AG35-QuecOpen

ECALL API MANUAL

LTE Module Series

Rev. AG35-QuecOpen_ECALL_API_guide_manual_V1.5

Date: 2018-08-08

Status: Temporary

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
7th Floor, Hongye Building, No.1801 Hongmei Road, Xuhui District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://quectel.com/support/sales.htm

For technical support, or to report documentation errors, please visit:
http://quectel.com/support/technical.htm
Or email to: support@quectel.com

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT

About the Document

History

Revision	Date	Author	Description
1.0	2018-02-28	Laurence YIN	Initial
1.1	2018-03-17	Laurence YIN	Modification
1.2	2018-03-27	Laurence YIN	Updated
1.3	2018-04-04	Laurence YIN	Updated
1.4	2018-05-31	Laurence YIN	Updated Push API
1.5	2018-08-08	Laurence Yin	Update QL_Voice_Call_Ecall struct

Contents

Figure Index

1 Introduction

This document mainly introduces how to use the eCall function of Quectel standard module. eCall function is only supported by the special software version.

eCall is defined as a manually or automatically initiated emergency call from a vehicle, supplemented with a minimum set of emergency related data (MSD), as defined under the EU Commission's eSafety initiative. It can be depicted by the figure below.

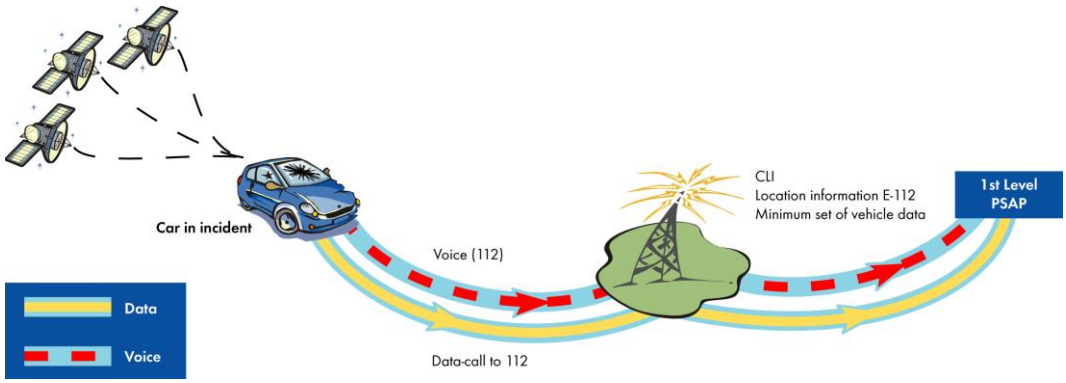

Figure 1: eCall System Overview

The architecture of eCall system is described in Figure 2. In Quectel test system, the module has the ability to act as IVS and also to simulate the PSAP. Thus, eCall testing can be easily performed by preparing two Quectel modules in the circumstance without access to a real PSAP. It will be described in the following chapters. Of course, if a real PSAP can be accessed, testing in the real environment is preferred.

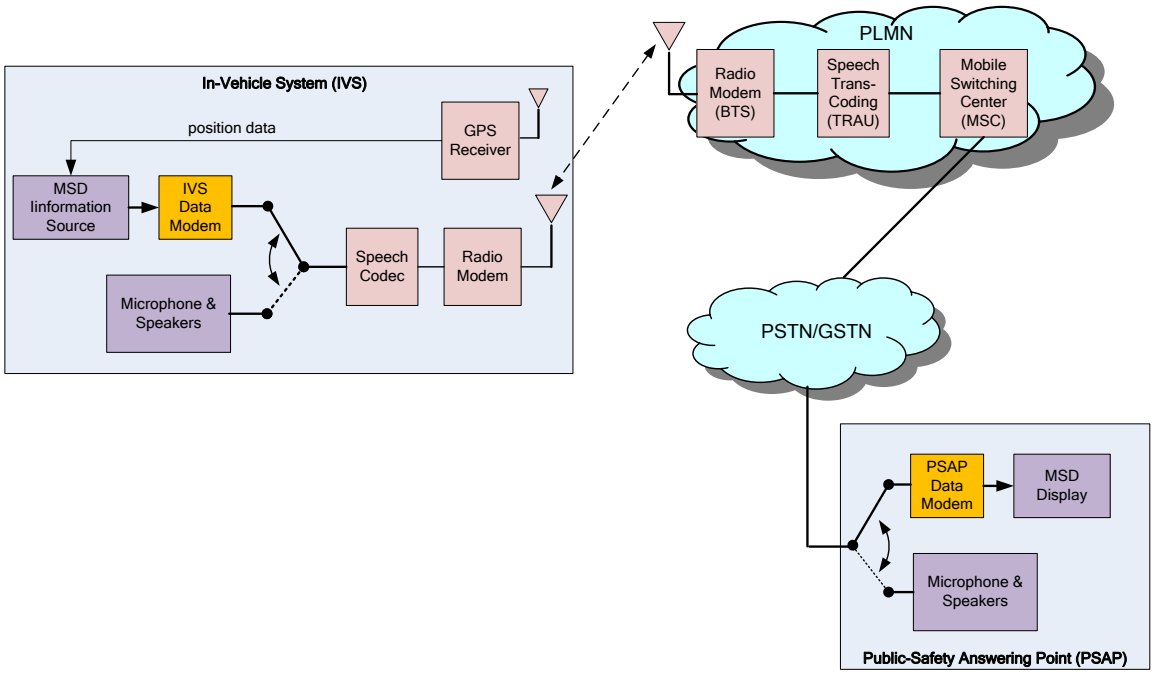

Figure 2: eCall System Architecture

2 ECALL API

2.1. QL_Voice_Call_Client_Init

(1) Function prototype:
int QL_Voice_Call_Client_Init(voice_client_handle_type *ph_voice);
(2) Parameter description:
1) ph_voice: OUT the pointer of voice handle
(3) Return description: int, 0-SUCCESS, Greater than 0- partial SUCCESS, Less than 0- FAILURE
(4) Functional description:
Init Voice function handle.

2.2. QL_Voice_Call_Client_Deinit

(1) Function prototype:
int QL_Voice_Call_Client_Deinit (voice_client_handle_type h_voice);
(2) Parameter description:
1) h_voice: IN voice handle
(3) Return description:int,0-SUCCESS, Greater than 0- partial SUCCESS,Less than 0- FAILURE
(4) Functional description:
Destroy related Voice feature resources

2.3. QL_Voice_Call_AddStateHandler

(1) Function prototype:
int QL_Voice_Call_AddStateHandler(voice_client_handle_type h_voice,
QL_VoiceCall_StateHandlerFunc_t handlerPtr,
void* contextPtr)
(2) Parameter description:
1) h_voice: IN voice handle
2) handlerPtr: IN voice callback function

3) contextPtr IN (the content of incoming call_id)

(3) Return description: int,0-SUCCESS, Greater than 0- partial SUCCESS, Less than 0- FAILURE

(4) Functional description:

Register the callback function to receive the incoming voice;

2.4. QL_Voice_Call_RemoveStateHandler

(1) Function prototype:

int QL_Voice_Call_RemoveStateHandler(voice_client_handle_type h_voice)

(2) Parameter description:

1) h_voice: IN voice handle

(3) Return description: int,0-SUCCESS, Greater than 0- partial SUCCESS, Less than 0- FAILURE

(4) Functional description:

Destroy the registered the callback function;

2.5. QL_Voice_Call_Ecall

(1) Function prototype:

```
int QL_Voice_Call_Ecall(voice_client_handle_type    h_voice,
                        E_QL_VCALL_ID_T             simId,
                        char*                       phone_number,
                        ql_mcm_ecall_info           ecall_info,
                        int                        *call_id);
```

(2) Parameter description:

1) h_voice: IN voice handle
2) simId: INslot ID（**dumped params**）
3) phone_number: IN The called number
4) ecall_info IN msd info
5) call_id: OUT call ID

(3) Return description: int,0-SUCCESS, Greater than 0- partial SUCCESS, Less than 0- FAILURE

(4) Functional description:

Originating-caller make a ecall；

2.6. QL_Voice_Call_Ecall_HangUp

(1) Function prototype:

int QL_Voice_Call_Ecall_HangUp (voice_client_handle_type h_voice)

(2) Parameter description:

1) h_voice: IN voice handle

(3) Return description: int,0-SUCCESS, Greater than 0- partial SUCCESS,Less than 0- FAILURE

(4) Functional description:

Hang up；

2.7. QL_Voice_Call_AddCommonStateHandler

(1) Function prototype:

```
int QL_Voice_Call_AddCommonStateHandler(voice_client_handle_type h_voice,
                                        QL_VoiceCall_CommonStateHandlerFunc_t handlerPtr);
```

(2) Parameter description:

1) h_voice: IN voice handle
2) handlerPtr IN

(3) Return description: int,0-SUCCESS, Greater than 0- partial SUCCESS, Less than 0- FAILURE

(4) Functional description:

Register the callback function to receive the ind info;

2.8. QL_Voice_Call_RemoveCommonStateHandler

(1) Function prototype:

```
int QL_Voice_Call_RemoveCommonStateHandler(voice_client_handle_type h_voice);
```

(2) Parameter description:

1) h_voice: IN voice handle

(3) Return description: int,0-SUCCESS, Greater than 0- partial SUCCESS, Less than 0- FAILURE

(4) Functional description:

Destroy the registered the callback function;

2.9. QL_Voice_Call_Ecall_MsdPush

(1) Function prototype:

```
int QL_Voice_Call_Ecall_MsdPush
(
    voice_client_handle_type    h_voice,
    E_QL_MCM_ECALL_STATE_T *ecall_state
);
```

(2) Parameter description:

1) h_voice: IN voice handle

2) ecall_state: OUT ecall voice satus

(3) Return description: int,0-SUCCESS, Greater than 0- partial SUCCESS, Less than 0- FAILURE

(4) Functional description:

IVS push api;

2.10. QL_Voice_Call_Ecall_UpdateMsd

(1) Function prototype:

```
int QL_Voice_Call_Ecall_UpdateMsd
(
    voice_client_handle_type    h_voice,
    const char *hex_msd
);
```

(2) Parameter description:

1) h_voice: IN voice handle

2) hex_msd:IN msd hex string

(3) Return description: int,0-SUCCESS, Greater than 0- partial SUCCESS, Less than 0- FAILURE

(4) Functional description:

Update IVS msd;

3 Program Steps Of The Demo

Please refer to **example/ecall/example_ecall.c**
Description:

step1:QL_Voice_Call_Client_Init------ register voice client
step2:QL_Voice_Call_AddCommonStateHandler ----- register callback
step3: Communication
step4:QL_Voice_Call_RemoveCommonStateHandler ----destroy callback
step5:QL_Voice_Call_Client_Deinit----- destroy client

4 Execution of the demo

4.1. Execute the command

```
/usrdata # ./example_ecall
```

4.2. MO ECALL（E_QL_MCM_ECALL_TEST 模式）

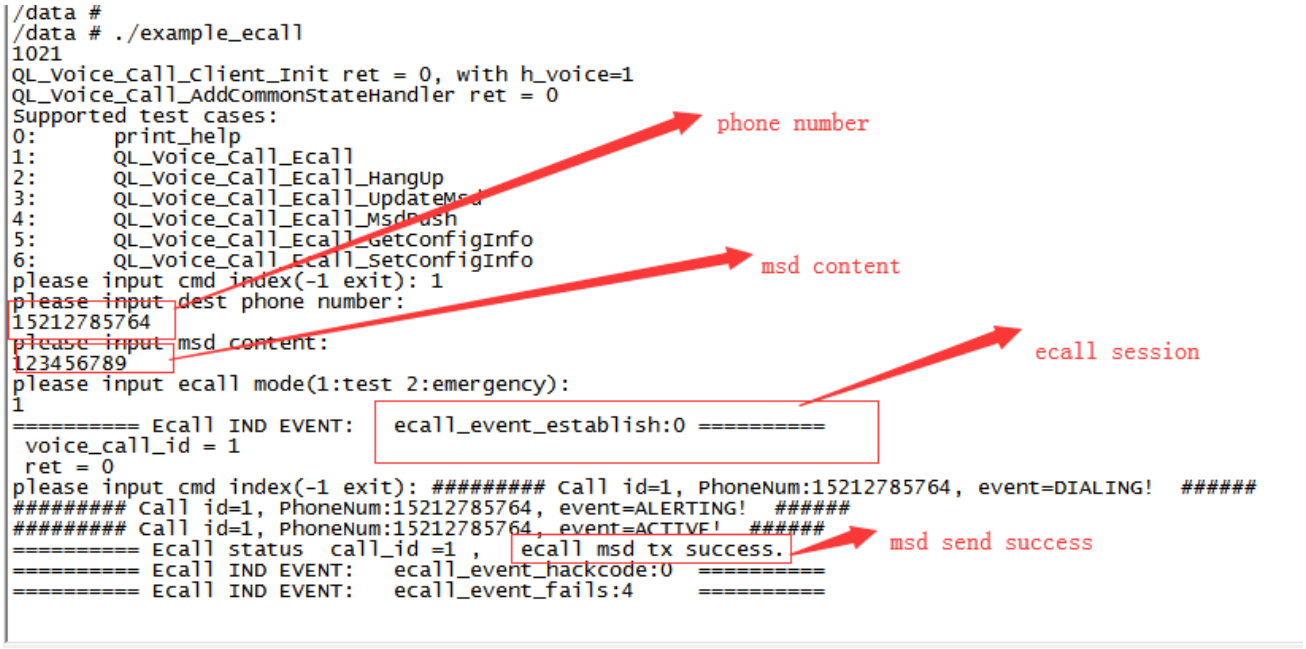

图 1:IVS MO ecall

The IVS terminal initiated ECALL, and when the PSAP terminal answered, the IVS sent MSD automatically. After the MSD was successfully received by PSAP, the event was reported to the IVS terminal.

```
/data # ./example_ecall
1021
QL_Voice_Call_Client_Init ret = 0, with h_voice=1
QL_Voice_Call_AddCommonStateHandler ret = 0
Supported test cases:
0:      print_help
1:      QL_Voice_Call_Ecall
2:      QL_Voice_Call_Ecall_HangUp
3:      QL_Voice_Call_Ecall_UpdateMsd
4:      QL_Voice_Call_Ecall_MsdPush
5:      QL_Voice_Call_Ecall_GetConfigInfo
6:      QL_Voice_Call_Ecall_SetConfigInfo
please input cmd index(-1 exit): 1
please input dest phone number:
15212785764
please input msd content:
123456789
please input ecall mode(1:test 2:emergency):
1
========== Ecall IND EVENT:   ecall_event_establish:0 ==========
 voice_call_id = 1
 ret = 0
please input cmd index(-1 exit): ######### Call id=1, PhoneNum:15212785764, event=DIALING!  ######
######### Call id=1, PhoneNum:15212785764, event=ALERTING!  ######
######### Call id=1, PhoneNum:15212785764, event=ACTIVE!  ######
========== Ecall status  call_id =1 ,   ecall msd tx success.
========== Ecall IND EVENT:   ecall_event_hackcode:0  ==========
========== Ecall IND EVENT:   ecall_event_fails:4     ==========
```

call status

图 2：IVS received event

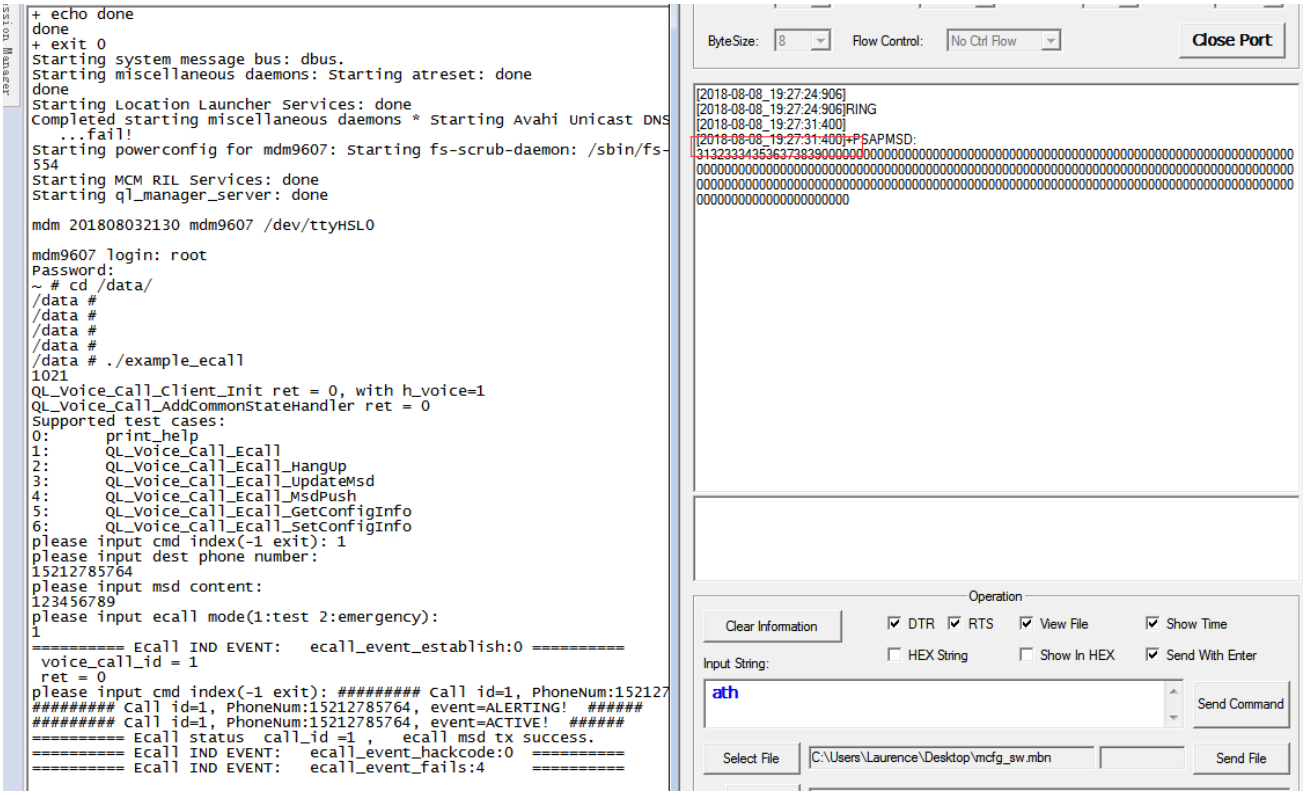

图 3 PSAP dispay MSD

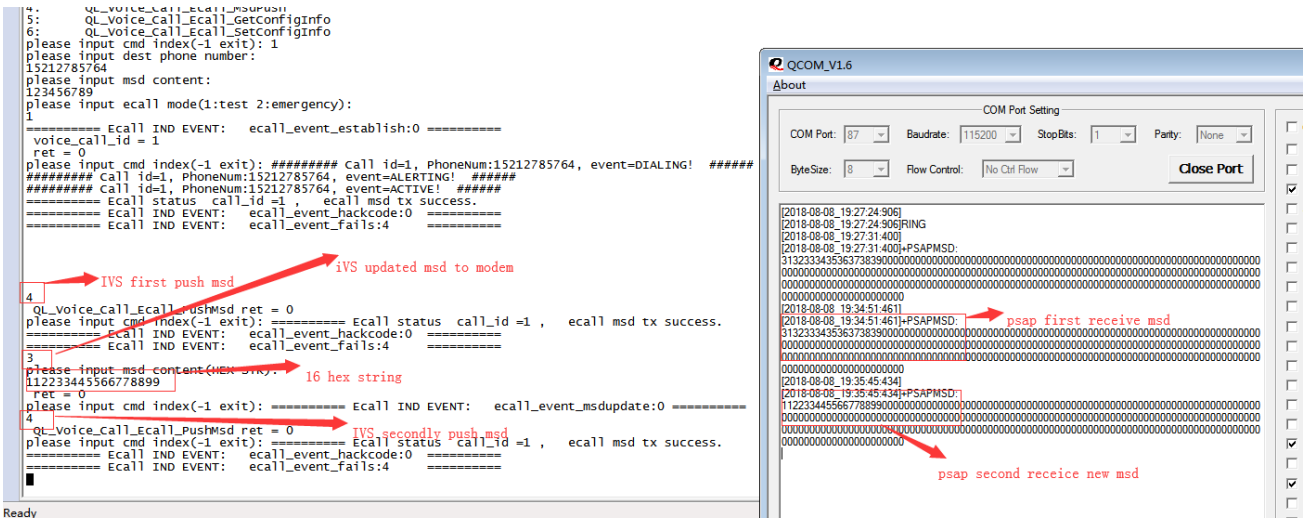

图 4 IVS PUSH MODE

When IVS and PSAP make a voice call, they can use the PUSH command to send a new MSD again.

4.3. MT ECALL

When eCall has established and PSAP terminates the eCall, MT eCall will be valid in 12 hours. For MT eCall, when eCall is established, MSD can be transferred in push mode or pull mode.

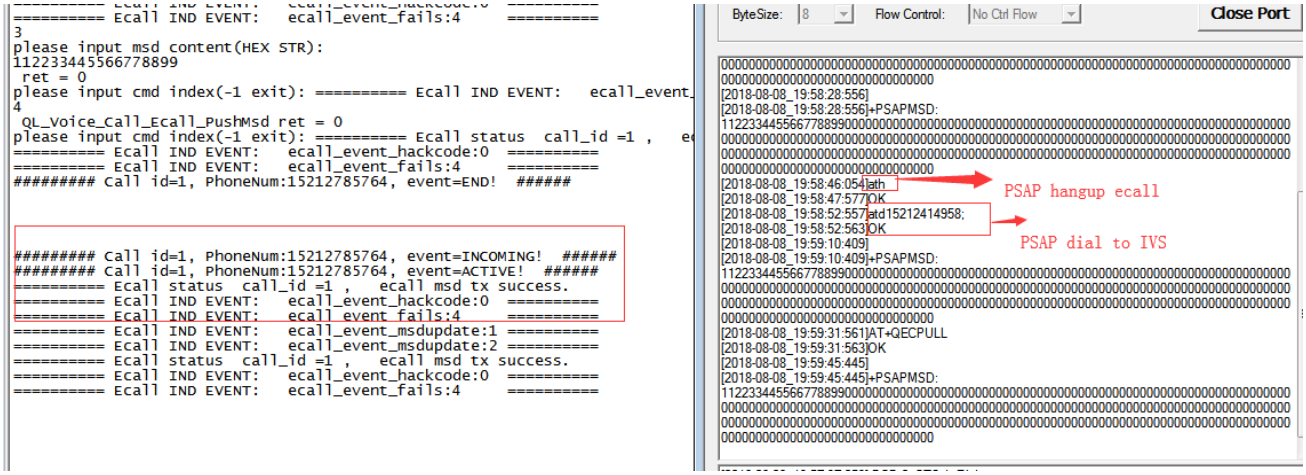

图 5 PSAP ORI CALL

Figure 5. The PSAP MO a call to IVS, and IVS began to send MSD after answer the call automatically. After successful transmission, ivs can make voice call with psap.

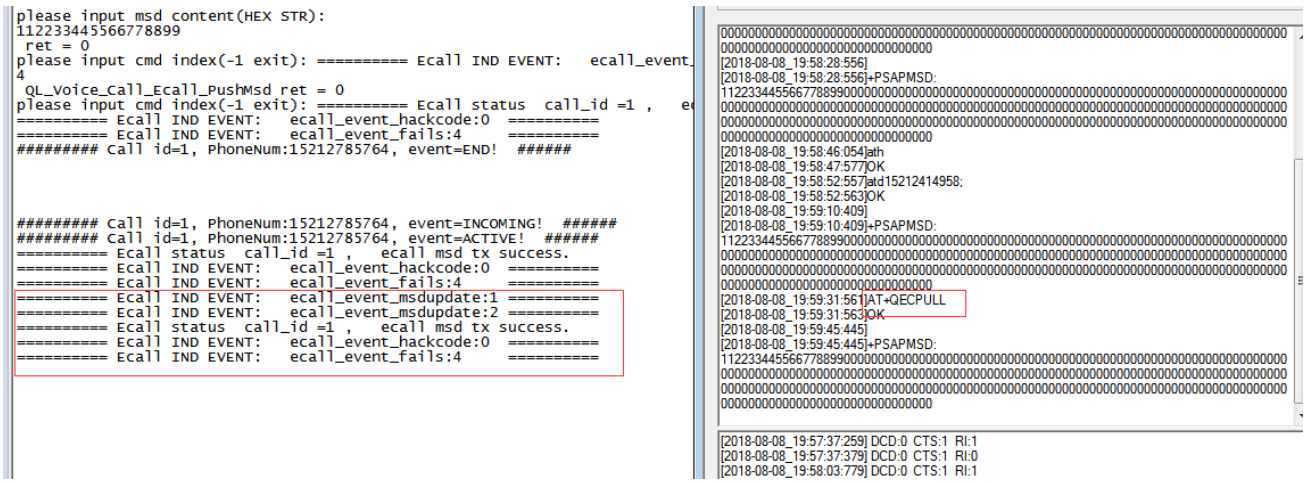

图 6 PSAP PULL mode

Figure 6 PSAP PULL mode, PSAP can send the PULL command to request the IVS side to send MSD content.

5 eCall Compiling Introduction

This chapter is the introductions of compiling single example_ecall.c.

1. Unzip tar -jxvf ql-ol-sdk.tar.bz2: tar -jxvf ql-ol-sdk.tar.bz2
2. Enter ql-ol-sdk: cd cd ql-ol-sdk
3. EnsureSDK version is same as firmware version: source ql-ol-crosstool/ql-ol-crosstool-env-init
4. Execute: cd ql-ol-extsdk/example/ecall
5. Execute: make clean;make