# Proactive Command知识总结

|  |  |  |
| --- | --- | --- |
| Date  日期 | Version  版本 | Comments  备注 |
| 2019-03-20 | 0.1 | First version |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Function  职位 | Name  姓名 | Date  日期 | Signature  签名 |
| Written by  拟定 | SW | 曾灿炫 | 2019-03-20 | 曾灿炫 |
| Verified by  审核 |  |  |  |  |
| Verified by  审核 |  |  |  |  |
| Approved by  批准 |  |  |  |  |

目录

[Proactive Command知识总结 1](#_Toc4180514)

[1. 概述 3](#_Toc4180515)

[**1.1.** **本文档的目的** 3](#_Toc4180516)

[**1.2.** **背景** 3](#_Toc4180517)

[**1.3.** **缩略语清单** 3](#_Toc4180518)

[**1.4.** **参考文献** 3](#_Toc4180519)

[2. 主动式命令简介 4](#_Toc4180520)

[2.1. 定义 4](#_Toc4180521)

[2.2. 流程 4](#_Toc4180522)

[3. Modem端简述 5](#_Toc4180523)

[3.1. 涉及模块 5](#_Toc4180524)

[3.2. 主要流程 5](#_Toc4180525)

[4. 例子分析 6](#_Toc4180526)

[4.1 代码跟进与log分析 6](#_Toc4180527)

[4.1.1 关键log 6](#_Toc4180528)

[4.1.2 分析 7](#_Toc4180529)

[4.2 时序图 9](#_Toc4180530)

[5. 经验分享 9](#_Toc4180531)

[6. 总结 10](#_Toc4180532)

# 概述

* 1. **本文档的目的**

*本文档作为一个高通modem的学习输出，总结STK模块的Proactive Command在Modem的运作流程。*

* 1. **背景**

*STK即SIM Tool Kit，它提供一系列用于移动设备与SIM卡间交互的机制。通过这些机制，支持STK的手机可以操作SIM卡里的应用。主动式命令便是其中之一。*

*大部分主动式命令实在应用层（AP端）进行处理，而与SIM卡直接接触的是BP端，即Modem，平时解决问题也少不了接触Modem端的代码，学习了解Modem端的处理机制能提高实际工作的效率。*

* 1. **缩略语清单**

*列出文中使用的术语的定义和缩略（语）词的英文全名和中文解释。*

| Term | Explanation |
| --- | --- |
| TR | Terminal Response |
|  |  |
|  |  |

* 1. **参考文献**

| Document | Explanation |
| --- | --- |
| 80-ng610-1\_k\_gstk\_overview | 高通文档 |
| 80-nk374-1\_e\_mmgsdi\_overview | 高通文档 |
| 3GPP 31.111 | Universal Subscriber Identity Module (USIM)  Application Toolkit (USAT) |

# 主动式命令简介

## 定义

主动式命令是由主动式SIM卡发出，SIM卡通过主动式命令让UE端执行对应的操作，如插入联通卡后出现的弹框消息为主动式命令Display Text。

主动式命令有数十种，具体可查看3GPP TS 11.11。

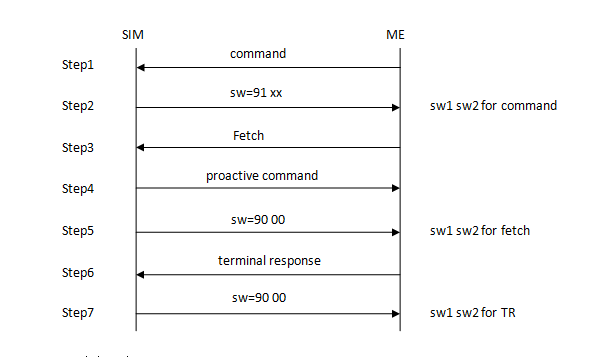
主动式SIM卡通过状态字SW1 SW2实现主动权，相关的状态字与解释如下：

|  |  |  |
| --- | --- | --- |
| SW1 | SW2 | 解释 |
| 90 | 00 | 正常的成功报告，SIM卡无其他SAT命令 |
| 91 | xx | 命令成功执行，有一个长为xx的proactive command |
| 93 | 00 | SIM卡正忙于执行前一个proactive command |
| 9E | xx | SIM数据下载时出错，有一个长为xx的回应 |

’90 00’表示命令正常结束。’91 XX’除了像’90 00’一样通知ME SIM卡成功执行完前一个命令，还包含了一个SIM卡的命令，告诉ME还需要执行一个动作。然后ME使用fetch命令来取得这个数据，XX表示相应数据的长度。

## 流程

一个完整的Proactive Command包括Fetch, Proactive Command, Terminal Response以及状态字，具体的时序图如下：



step1-7称为一个sat session.

1．ME总是命令的发起者，首先ME发送SAT命令给SIM卡

2． SIM卡执行完命令后，返回Status Words（91XX），告诉ME有一个proactive command需要执行

3．ME用fetch命令取得proactive command

4．SIM将proactive command数据传送给ME

5．SW1 SW2 for Fetch

6．ME发送 Terminal Response给SIM卡

7．SW1 SW2 for Terminal Response

Step4-6是ME执行proactive command的时间：如显示数据，让用户输入，播放声音，发送短信等等。

下文将根据这个图分析Proactive Command的处理流程。

# Modem端简述

## 涉及模块

**UIM:**负责卡相关的程序，也是卡与UE间的唯一通信接口，分为UIM driver和UIM server。driver能启动，重启卡，发送指令等；server面向mmgsdi、gstk等，其中一项作用便是支持STK相关的操作，如主动式命令流程中的fetch。参考文档为80-nh940-1\_f。

**MMGSDI(Multimode Generic SIM Driver Interface)：**提供通用库函数，供SIM卡调用，另一方面供上层检索、修改SIM卡的数据。

**GSTK(Generic SIM Application Toolkit）：**负责STK相关流程，发送TP，解析来自SIM卡的主动式命令和Envelope response，向外提供发送TR和Envelop命令的库函数。参考文档为80-ng610-1\_k。

**QMI CAT：**高通项目中RIL层没有与Modem直接通信，而是通过中间层QMI+QCRIL实现消息的传递与方法调用，QMI (Qualcomm MSM™ Interface )分为QMI UIM和QMI CAT(QMI Card Application Toolkit)，负责STK业务的为QMI CAT，主要文件为qmi\_cat.c。

## 主要流程

一个主动式命令的主要流程如下图，自下向上依次涉及的模块有UIM、GSTK与MMGSDI、QMI\_CAT。



# 例子分析

以set up menu为例

## 4.1 代码跟进与log分析

### 4.1.1 关键log

以Benz手机为例，插入卡抓取set up menu的modem log，截取主要部分如下：

**Fetch下发：**

1.[uim.c 1795] UIM\_1: cmd status 0x0 SW1 0x91,SW2 0x66, Response data length 0x32

2.[uimgen.c 3453] UIM\_1: Reporting UIM generic command 0x104

3.[gstklib.c 3525] MMGSDI FETCH allow Status is 0x1

4.[gstklib.c 3528] Setting proactive\_cmd\_in\_progress flag to 0x1

5.[uimgen.c 2737] UIM\_1: Received Internal Fetch command 0x111

6.[uimdrv.c 2391] UIM\_1: uim\_send\_command

7.[gstklib.c 5368] gstk\_gsdi\_set\_fetch\_status: Slot = 0x1, status = 0x1

**获取主动式命令：**

8.[uim.c 1795] UIM\_1: cmd status 0x0 SW1 0x90,SW2 0x0, Response data length 0x1c

9.[uimgen.c 3453] UIM\_1: Reporting UIM generic command 0x111

10.[gstklib.c 2734] UI or NW proactive session in progress 0xff

11.[gstk\_s\_idle.c 686] Received Proactive cmd

12.[gstk\_proactive\_cmd.c 878] In gstk\_process\_proactive\_command() ui\_nw\_slot\_in\_progress = 0xff, cmd slot\_id 0x1

[gstk\_proactive\_cmd.c 927] Cache the proactive cmd

[gstk\_proactive\_cmd.c 598] COMMAND DETAILS CMD NUM: 0x1, TYPE OF CMD: 0x25, QUALIFIER: 0x0

13.[qmi\_cat.c 6204] qmi\_cat\_cache\_raw\_proactive\_command: command\_id=0x25, slot\_index=0x

[qmi\_cat.c 12608] Sending QMI CAT Indication to QMI Framework: clid:0x1, ind\_id:0x1

**发送Terminal Response:**

14. [uim.c 1795] UIM\_1: cmd status 0x0 SW1 0x90,SW2 0x0, Response data length 0x1c

15. [ gstk\_terminal\_rsp.c 2872] IN gstk\_send\_raw\_terminal\_response

16. [gstkutil.c 7722] 0x81 0x3 0x1 0x25 0x0 0x2 0x2 0x82 0x81

17. [uimgen.c 2750] UIM\_1: Received Terminal Response command 0x112

18. [uimdrv.c 2391] UIM\_1: uim\_send\_command

19. [uim.c 1795] UIM\_1: cmd status 0x0 SW1 0x90,SW2 0x0, Response data length 0x0

20. [uimgen.c 3453] UIM\_1: Reporting UIM generic command 0x112

### 4.1.2 分析

根据4.1.1所列序号对log挨个分析：

**Fetch下发：**

1. uim\_command\_response处理SIM卡上报的response,读取返回状态字为SW1SW2:9166，即上一条命令执行成功，有一条长度0x66的主动式命令。
2. 由uim\_generic\_command\_response确定返回的是命令0x104的执行结果。
3. uim.c调用uim\_process\_card\_response进行处理，通过uim\_command\_response解读response确定stauts为fetch，调用uim\_interface\_gstk\_uim\_fetch\_allow确定是否能发送fetch，uim\_interface\_gstk\_uim\_fetch\_allow实际调用了gstklib的gstk\_uim\_fetch\_allow：

case:UIM\_CMD\_FETCH

/\* Mimic the case of no-FETCH to UIM\_CMD\_COMPLETED case \*/

if(status == UIM\_CMD\_FETCH &&

!**uim\_interface\_gstk\_uim\_fetch\_allow**((gstk\_slot\_id\_enum\_type)cmd\_slot)) {

status = UIM\_CMD\_COMPLETED;

}

1. 进入gstklib的gstk\_uim\_fetch\_allow，根据注释解释该函数提供给UIM去确定能否发fetch，这里返回1，不改变状态值；继而调用GSTK\_SET\_PROACTIVE\_CMD\_IN\_PROGRESS\_FLAG设置标志表示主动式命令处理中。

/\* Mimic the case of no-FETCH to UIM\_CMD\_COMPLETED case \*/

if(status == UIM\_CMD\_FETCH &&

!uim\_interface\_gstk\_uim\_fetch\_allow((gstk\_slot\_id\_enum\_type)cmd\_slot))

{

status = UIM\_CMD\_COMPLETED;

}

1. 根据状态值仍为UIM\_CMD\_FETCH，确定执行fetch命令：

/\* else if a proactive command is pending \*/

else if (status == UIM\_CMD\_FETCH)

{

/\* Send a FETCH command to the SIM \*/

**uim\_fetch (i\_mask, uim\_ptr);**

}

1. 跟进uim\_fetch，最终调用了uimdrv.c的uim\_send\_command去发送fetch，即如上文所说通过UIM driver发送fetch命令：

case UIM\_FETCH\_ST: /\* Fetch State \*/

{

/\* Build a Fetch APDU \*/

uim\_req\_buf\_static\_ptr->apdu\_hdr.instrn = FETCH;

uim\_req\_buf\_static\_ptr->instrn\_case = UIM\_INSTRN\_CASE\_2;

/\* P1 = P2 = 0 and data field is empty in UICC implies that the

command applies on the current EF \*/

uim\_req\_buf\_static\_ptr->apdu\_hdr.p1 = 0;

uim\_req\_buf\_static\_ptr->apdu\_hdr.p2 = 0;

if (uim\_ptr->command.cmd\_ptr->hdr.protocol == UIM\_UICC)

{

uim\_req\_buf\_static\_ptr->apdu\_hdr.uim\_class = UIM\_UICC\_INSTRUCTION\_CLASS2;

}

else

{

uim\_req\_buf\_static\_ptr->apdu\_hdr.uim\_class = UIM\_INSTRUCTION\_CLASS;

}

/\* Number of bytes of Fetch command \*/

uim\_req\_buf\_static\_ptr->apdu\_hdr.p3 = uim\_ptr->command.cmd\_ptr->fetch.command\_size;

uim\_req\_buf\_static\_ptr->rsp\_ptr = &uim\_ptr->command.rsp\_buf;

uim\_send\_command(uim\_req\_buf\_static\_ptr, uim\_ptr);

} /\* end case - UIM\_FETCH\_ST \*/

break;

1. 由mmgsdi调用gstk\_gsdi\_set\_fetch\_status设置当前允许执行fetch命令，这里是由回调事件MMGSDI\_SESSION\_CHANGED\_EVT触发。

**获取主动式命令：**

1. uim\_command\_response处理SIM卡上报的response,读取返回状态字为9000，上一条命令执行成功。
2. 返回的是命令0x111的执行结果，根据uim\_cmd\_name\_type结构体的定义，0x111为UIM\_INTERNAL\_FETCH\_F，即fetch执行成功，开始主动式命令的处理。
3. 调用gstklib.c的gstk\_send\_proactive\_cmd将命令传至gstk。
4. gstk当前为空闲模式，调用gstk\_process\_proactive\_command开始处理主动式命令，
5. 进入gstk\_process\_proactive\_command ，确定是新的主动式命令，调用gstk\_preprocess\_proactive\_cmd进行解析，确定命令类型为0x25，即set up menu。

由gstk\_setup\_menu.c将命令发至客户端（qmi\_cat）：

/\* Send command to clients \*/

gstk\_status = gstk\_util\_send\_message\_to\_clients(&request);

1. 由qmi\_cat.将主动式命令发至上层，这里的Indication我理解为由SIM卡发起的命令，在RIL层由RadioIndication.java处理。

**发送Terminal Response:**

1. 主动式命令发送结束。

15-18. 发送Terminal Respone， 类型为0x25，即set up menu的Terminal Response。

19-20. Terminal Reponse发送完成，0x112即Terminal Reponse。

## 4.2 时序图

下图为set up menu上报的过程（不包含发送TR）



# 5. 经验分享

Defect：7098243

Summary: [STK]STK cannot enter when first time hot insert the SIM card in to ME

问题分析：

1. 第一次开机后无法进入stk应用，这种情况首先可以确定是set up menu相关，通过logcat可查看RILJ以及应用层的log，发现没有任何主动式命令相关的log，确定问题出现在底层。
2. 接下来就可根据上文所提的关键log进行排查，通过QXDM发现modem端正常执行了fetch下发，获取主动式命令的过程。
3. 在qmi\_cat.c没有预见的log（第13点），而是出现了异常的log: [ qmi\_cat.c 7817] QMI-CAT: No client to send indication: command\_id=0x11。 对比正常机发现nv项65683值错误，正常值应为2：[ qmi\_cat.c 12425] qmi\_cati\_get\_cached\_proactive\_cmd: config\_mode=0x3即问题出现在nv65683（qmi\_cat\_mode）。

排错：

1. 跟进代码确定qmi\_cat\_mode在qmi初始化时通过qmi\_cati\_read\_configuration读值。
2. 查看配置文件/nv/item\_files/modem/qmi/cat/qmi\_cat\_mode发现对应值为2，确定是读值失败。
3. 查看qmi\_cati\_read\_configuration确定读值失败情况下返回0x03：

return CATI\_CONFIG\_MODE\_DECODED;该值为0x03

# 总结

本文总结了Proactive Comment的知识点，通过协议文档简述其基础知识点，结合高通modem的代码与log介绍主动式命令的运作流程。

涉及主动式命令的问题大多与modem相关，了解主动式命令的处理过程与参与模块能保证及时高效的解决问题。