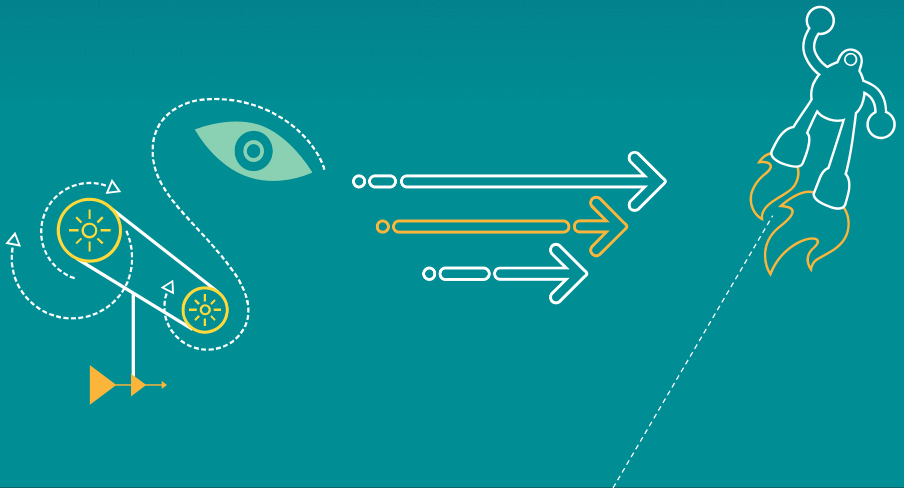
QMI UIM and QMI CAT Overview

Qualcomm Technologies, Inc.



80-NJ897-1 C

Confidential and Proprietary – Qualcomm Technologies, Inc.

Restricted Distribution: Not to be distributed to anyone who is not an employee of either Qualcomm Technologies, Inc. or its affiliated companies without the express approval of Qualcomm Configuration Management.

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision History

Revision	Date	Description
A	November 2013	Initial release
B	March 2015	Enhanced the security for PIN encryption.
C	November 2016	Added slides 25 and 26.

Contents

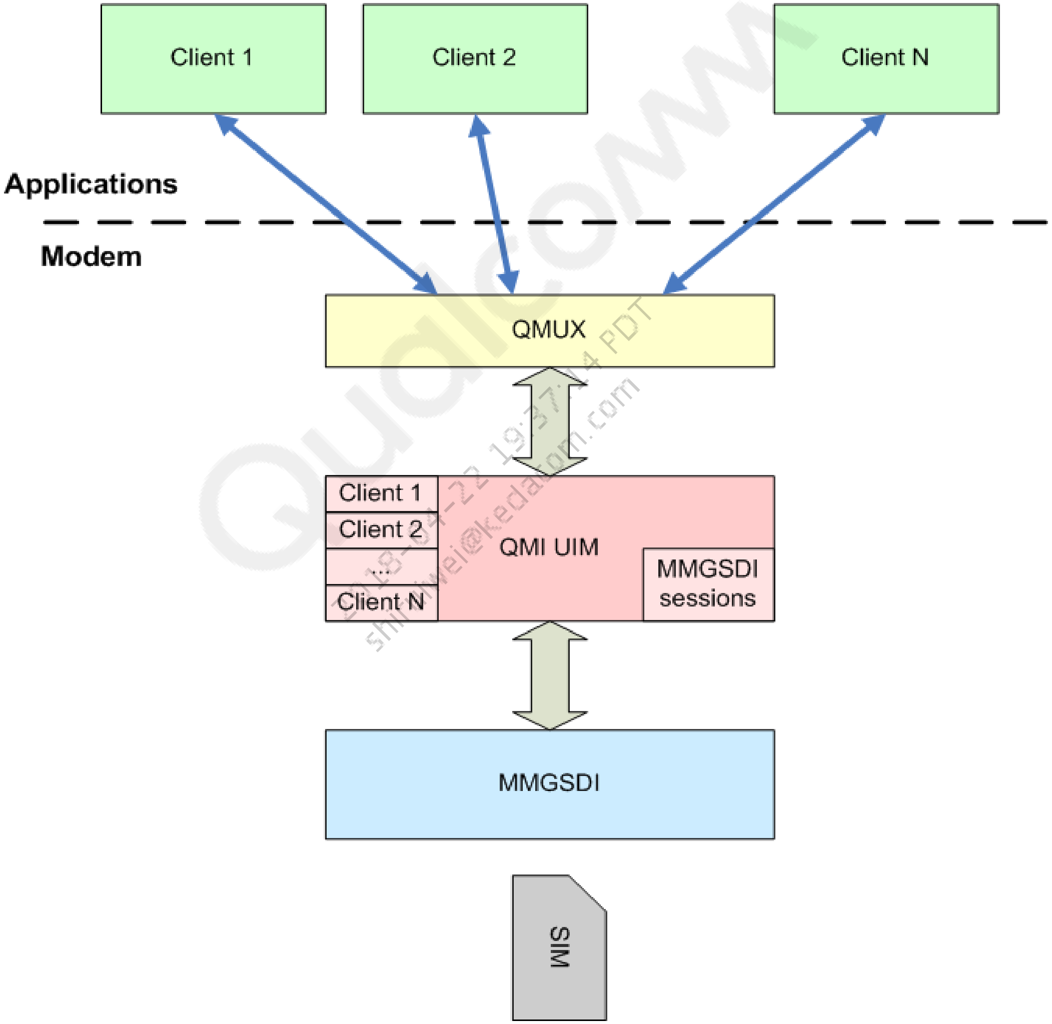
- QMI UIM
  - QMI UIM Description
  - QMI UIM Architecture
  - QMI UIM Commands
    - File Access
    - Encrypted PIN1
    - Refresh
    - Card Status
    - Subscription OK – Personalization on the Client
    - QMI_UIM_SET_APDU_BEHAVIOR
- QMI CAT
  - QMI CAT Description
  - QMI CAT Architecture
  - Call Flows
- References
- Questions?

QMI UIM

QMI UIM Description

- Qualcomm Messaging Interface (QMI) for the User Identity Module (QMI UIM) is one of the QMI services.
- It uses the QMI Multiplexing Protocol (QMUX) to communicate with clients.
  - Clients can run on the application processor
  - Clients can run on a PC connected to the device (QMI Testpro)
- Service ID is 0x0B
- QMI UIM service complies with the generalized QMI service specification, including the rules for messages, indications and responses, byte ordering, arbitration, constants, result, and error code values.
- Supports multiple requests from one client at the same time
- Supports multiple clients on the application side at the same time
  - For each client, QMI UIM stores a list of parameters:
    - Registered or not registered for events
    - List of files for refresh
    - Pending requests
- On Android™/Windows® Mobile, one client is the RIL layer
  - A thin library is developed to send/receive QMI requests/responses

QMI UIM Architecture



Provisioning Sessions

- QMI UIM automatically opens six Multimode Generic SIM Driver Interface (MMGSDI) sessions at power-up:
  - Two primary sessions (one for GW and one for 1X)
  - Two secondary sessions (one for GW and one for 1X)
  - One for files under Master File (MF) in slot 1
  - One for files under MF in slot 2
- These sessions are opened in all cases, even if the device only supports one access technology or only one slot.
  - In these cases, some sessions are invalid and QMI UIM returns an error if the client tries to use them
- The same sessions are shared among all QMI UIM clients.
  - Clients do not need to explicitly open the sessions
  - Clients are not aware of sessions; they only need to specify the type of application on the card they want to access

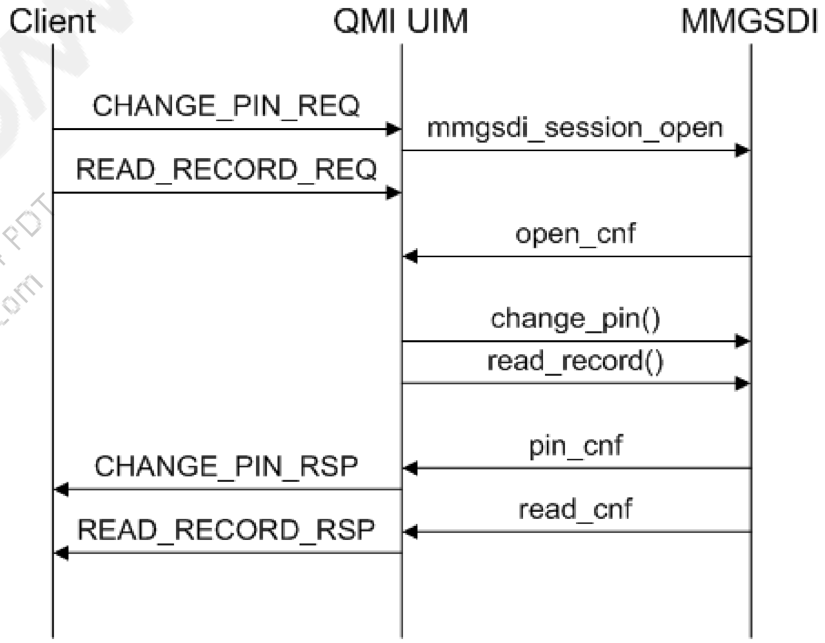
Nonprovisioning Sessions

- Access to nonprovisioning applications on the card is specified by clients passing the Application Identifier (AID) of the application in the request to QMI UIM.
- Nonprovisioning sessions are opened on the fly—only when they are needed.
  - The first command accessing a nonprovisioning application opens it.
  - It remains open until a client issues a QMI command to close it.
- Nonprovisioning sessions are shared among all QMI UIM clients in the same way as provisioning sessions.

Nonprovisioning Sessions – Call Flow

Successful case

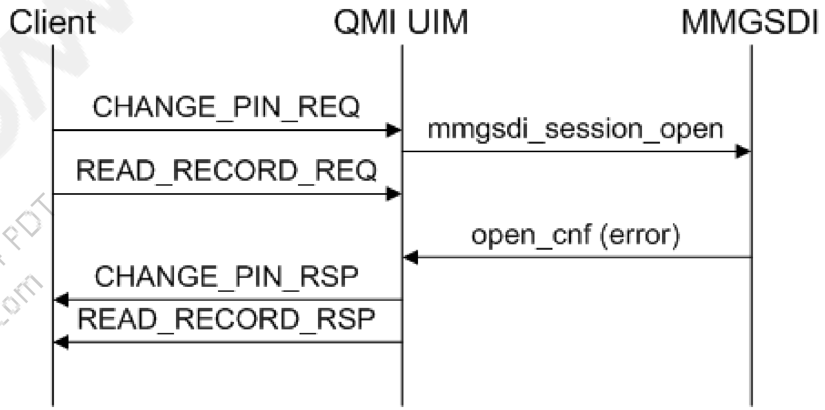
- Client makes one or more requests for a nonprovisioning application on the card
- The request is queued and QMI UIM opens the nonprovisioning session
- When the session is successfully opened, the request is sent to the modem
- QMI UIM handles the nonprovisioning sessions internally without the client being aware of the session



Nonprovisioning Sessions – Call Flow (cont.)

Error case

- Client makes one or more requests for a nonprovisioning application on the card
- The request is queued and QMI UIM opens the nonprovisioning session
- The session fails to open (many reasons are possible)
- All pending requests are terminated with an error response to the client



QMI UIM Commands

- File access
  - Read transparent and record
  - Write transparent and record
  - Get file attributes
- PIN operations
  - Verify PIN
  - Change PIN
  - Enable/disable PIN
  - Unblock PIN
- Refresh
  - Register for refresh
  - Vote for initialization
  - Refresh complete
  - Get last refresh event
- Others
  - Get card status
  - Personalization (only disable and unblock)
  - Power up and power down
  - Authenticate
  - Close nonprovisioning session

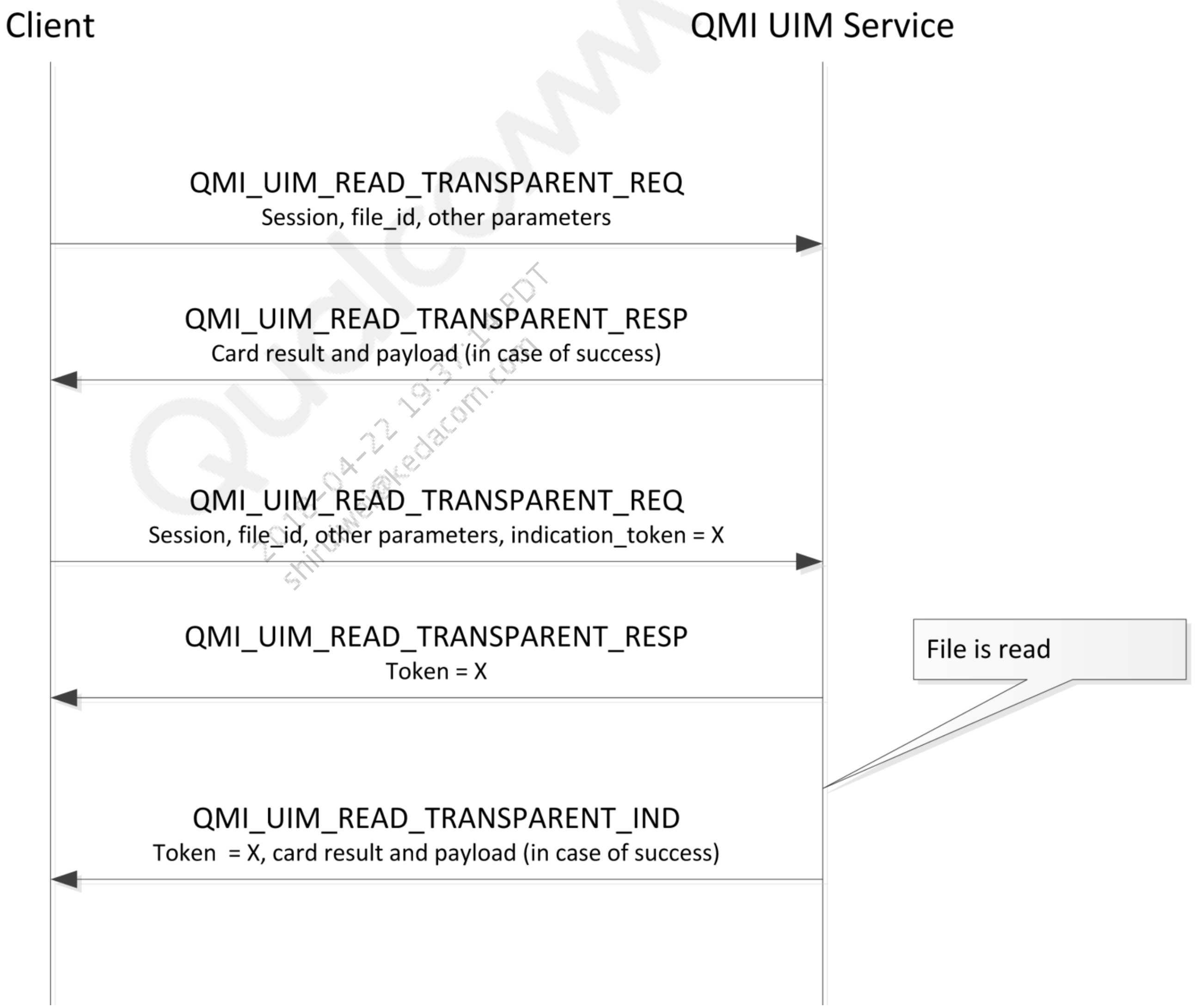
QMI UIM Parameters

- Sessions
  - All default sessions are identified only by the session type, which is passed as one of the parameters of the command
  - Nonprovisioning sessions are identified by the AID and the slot information
- Files
  - QMI UIM interface only supports access by path
  - Clients need to pass the complete path in all cases:
    - 3F00 2Fxx – Files under MF
    - 3F00 7F20 6Fxx – Files under GSF DF
    - 3F00 7FFF 6Fxx – Files in USIM or CSIM
  - This avoids exposing a long list with all supported files to QMI UIM clients (hard to maintain the backward compatibility)
  - QMI UIM has a mapping table to convert most common files into an MMGSDI enumeration, but it can proceed even if the file is not present in the table

File Access

- Operations that require access to the SIM card can experience a delay due to the interface between the device and the card.
- These delays become even longer in some circumstances, such as during power-up, when many card access requests are requested by many different clients.
- For this reason, an optional token can be specified in the command to request the modem to immediately send a response and provide the actual result later using a separate indication.
- The same token value is passed in both the response and in the subsequent indication so the client knows for which request the result is received.
- The token can be any value and can be used by the client to carry information to process the indication when the result is received.

File Access (cont.)



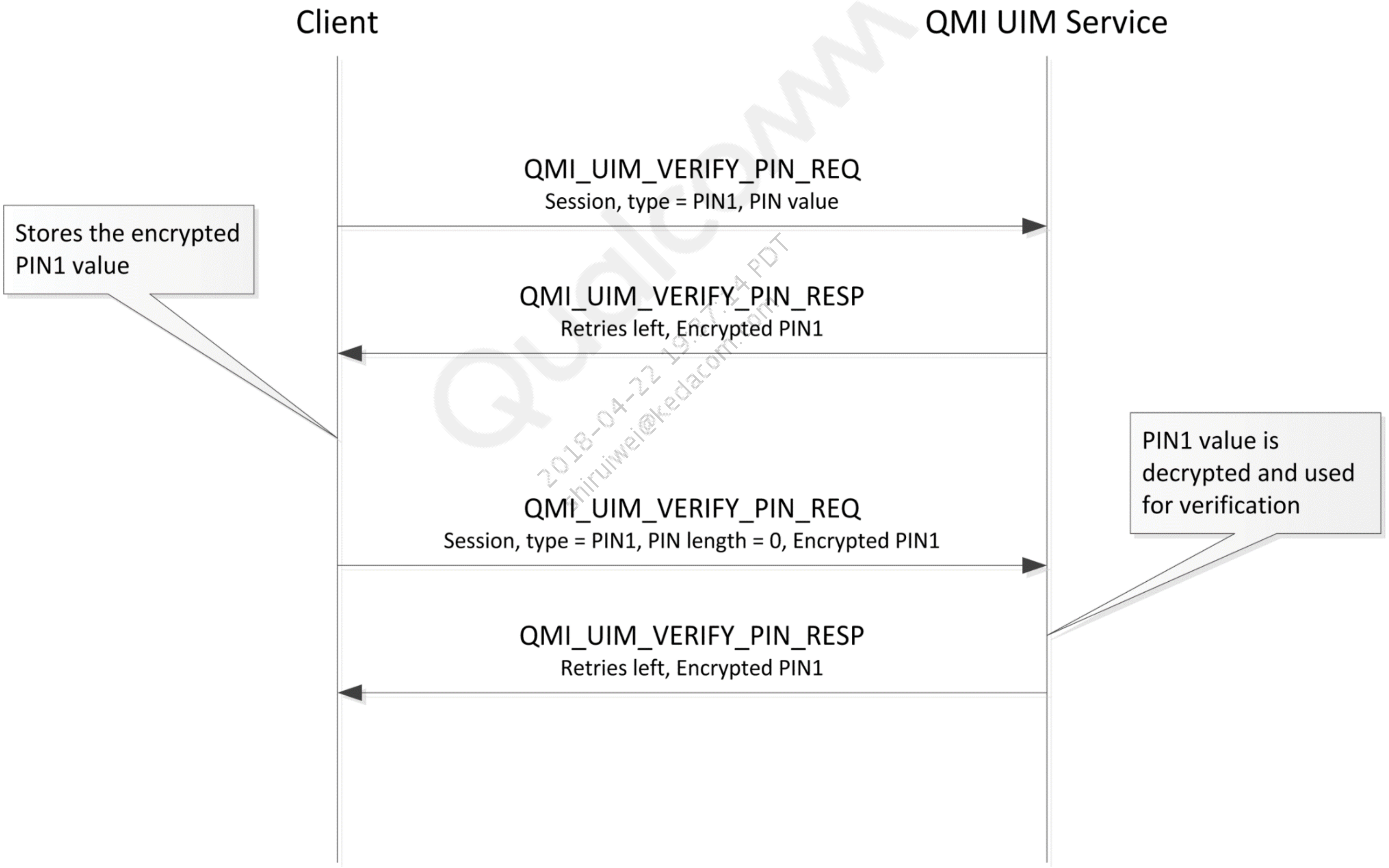
Client
QMI UIM Service
QMI_UIM_READ_TRANSPARENT_REQ
Session, file_id, other parameters
QMI_UIM_READ_TRANSPARENT_RESP
Card result and payload (in case of success)
QMI_UIM_READ_TRANSPARENT_REQ
Session, file_id, other parameters, indication_token = X
QMI_UIM_READ_TRANSPARENT_RESP
Token = X
File is read
QMI_UIM_READ_TRANSPARENT_IND
Token = X, card result and payload (in case of success)

Encrypted PIN1

- The QMI UIM service provides support for PIN1 encryption to implement features such as silent PIN verification after a restart of the modem. After any successful PIN operation (that is, verification, enablement, disablement, change, or unblock) is executed on PIN1, the QMI UIM service returns an encrypted value of PIN1 to the client.
- The encrypted value can be stored by the client and reused later to verify the PIN1 again. The encrypted PIN1 value cannot be used for any other PIN operation.
- Although the client has the encrypted value of PIN1, it does not have the key to decrypt it. While the modem has the key, it does not store the encrypted value. This solution guarantees security against attacks performed on one side only.

Note: This feature might not always be enabled on the modem. The client must not assume availability and must check the response for any PIN operation to determine whether the encrypted PIN1 value is present.

Encrypted PIN1 (cont.)



Client
QMI UIM Service
QMI_UIM_VERIFY_PIN_REQ
Session, type = PIN1, PIN value
Stores the encrypted PIN1 value
QMI_UIM_VERIFY_PIN_RESP
Retries left, Encrypted PIN1
QMI_UIM_VERIFY_PIN_REQ
Session, type = PIN1, PIN length = 0, Encrypted PIN1
PIN1 value is decrypted and used for verification
QMI_UIM_VERIFY_PIN_RESP
Retries left, Encrypted PIN1

Encrypted PIN1 (cont.)

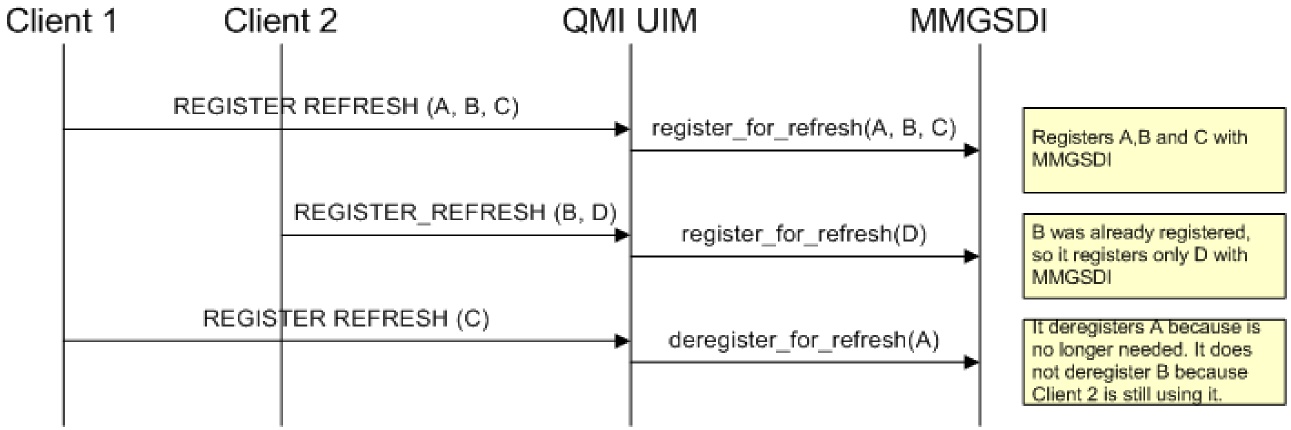
- To enhance the security of encrypted PIN1, the newer SECAPIs are used to encrypt and decrypt data after MPSS.TH.1.0, MPSS.TA.1.0, and later.
- The following QMI APIs are per the new approach (MPSS.TH.1.0, MPSS.TA.1.0, and later):
  - Messages
    - QMI_UIM_SET_PIN_PROTECTION
      - Response, only when PIN1 is enabled
      - Indication, only when IND is required and PIN1 is enabled
    - QMI_UIM_VERIFY_PIN
      - Request, when the encrypted PIN1 TLV is used
      - Response, only when PIN1 is verified successfully
      - Indication, only when IND is required and PIN1 is verified successfully
    - QMI_UIM_UNBLOCK_PIN
      - Response, only when PIN1 is unblocked successfully
      - Indication, only when IND is required and PIN1 is unblocked successfully
    - QMI_UIM_CHANGE_PIN
      - Response, only when PIN1 is changed successfully
      - Indication, only when IND is required and PIN1 is changed successfully
  - TLV – Encrypted PIN1

Encrypted PIN1 (cont.)

- Note that the Silent PIN1 feature must be enabled (NV item 66036 set to 1) for the encrypted TLV to be sent.
- There is no impact on the QMI UIM clients; the change in QMI UIM is backward compatible.
  - Even with the new SECAPIs, the encrypted PIN1 TLV is still able to hold the new data payload with the additional information (currently defined as 255 bytes)
  - Encrypted PIN1 + IV + MAC is < 255 bytes
- Final encrypted TLV data is sent as follows:
  - [16 bytes encrypted PIN1][16 bytes IV][32 bytes MAC]

Refresh – Registration

- QMI UIM creates a single list of all files registered for refresh.
  - This logic is required to share the same session among all QMI clients.
  - The list is updated every time a client sends a new REGISTER_REFRESH request.
  - For each client QMI UIM keeps a list of files that it registered for refresh, to send the notification only to the correct clients.
- Every client can specify a different list for each session type.
- QMI UIM supports up to two refresh procedures in parallel, one for each slot.
  - This might not be supported by the modem.



Client 1

Client 2

QMI UIM

MMGSDI

REGISTER REFRESH (A, B, C)

register_for_refresh(A, B, C)

Registers A,B and C with MMGSDI

REGISTER_REFRESH (B, D)

register_for_refresh(D)

B was already registered, so it registers only D with MMGSDI

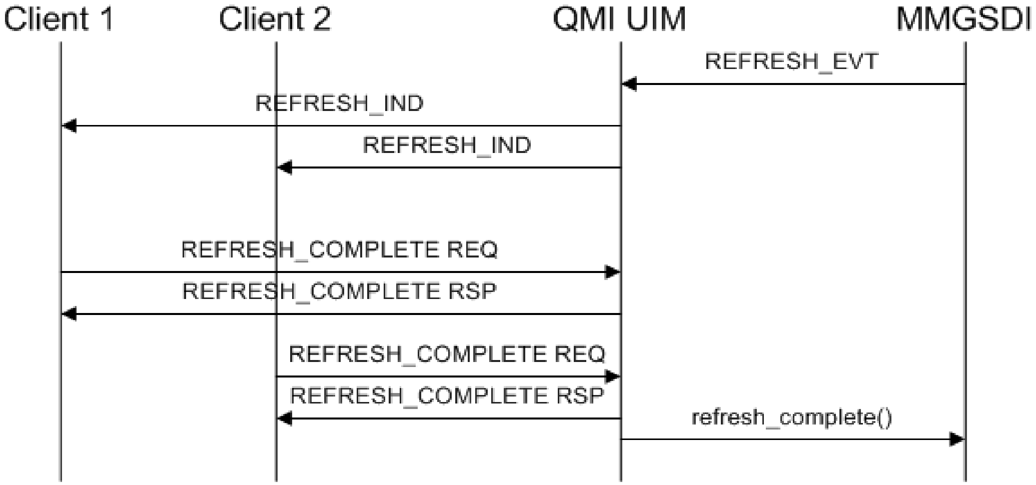
REGISTER REFRESH (C)

deregister_for_refresh(A)

It deregisters A because is no longer needed. It does not deregister B because Client 2 is still using it.

Refresh – Call Flow

- For the refresh call flow:
  - During the refresh, QMI UIM sends an indication to all registered clients.
  - It waits until every client completes the refresh.
  - At the end, QMI UIM sends refresh_complete() to MMGSDI.
- Each client can also decide if it wants to vote for initialization, in case of an INIT refresh.
- The mechanism and flows are the same used by MMGSDI.
  - Each MMGSDI event is converted into a REFRESH indication to QMI clients.
  - Each MMGSDI API is converted into a REFRESH request for QMI clients.
- If there is not any client interested in the refresh, QMI UIM automatically responds to it (vote for initialization, complete).



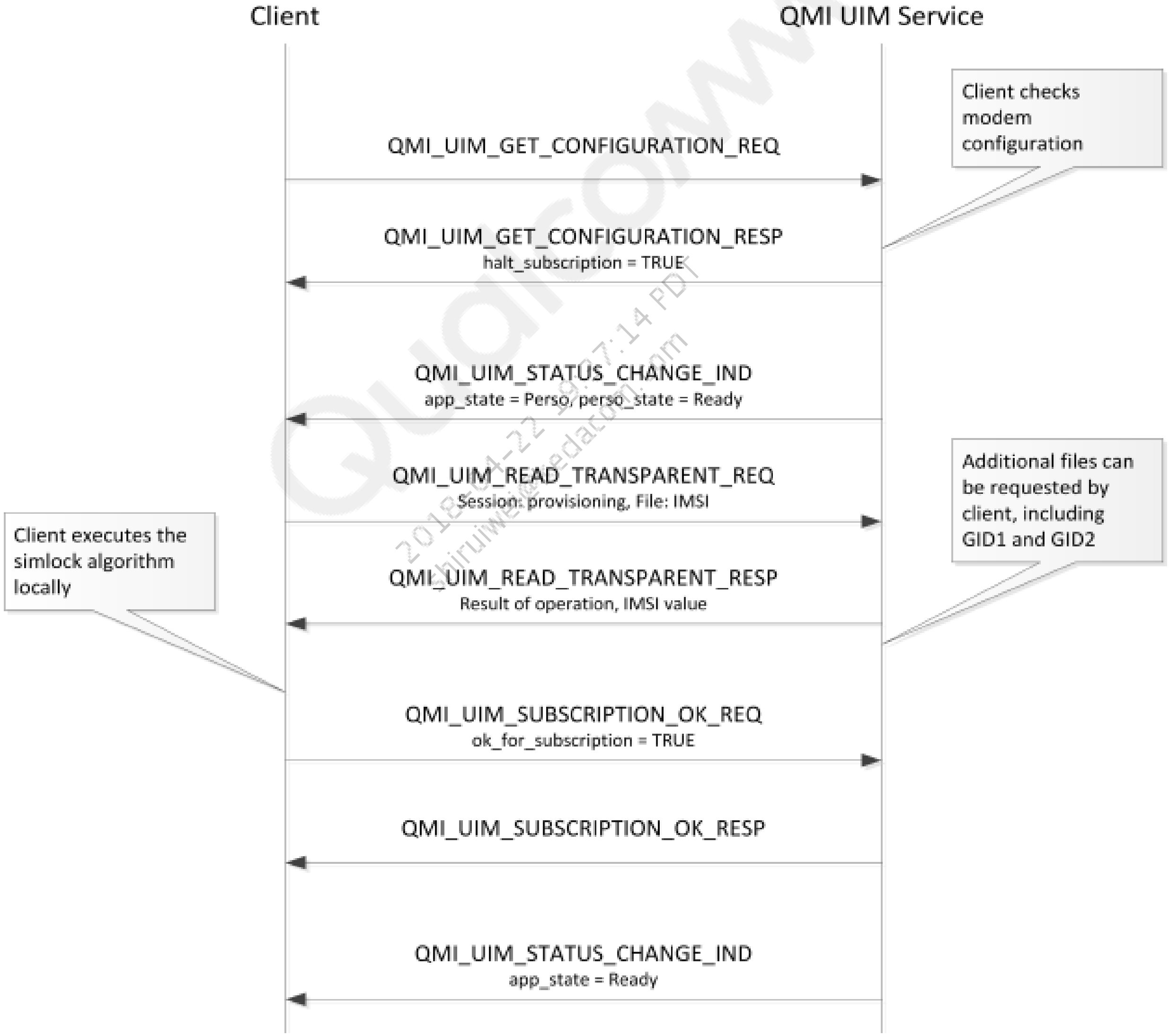
Card Status

- QMI UIM tracks the card status registering for events from MMGSDI
  - QMI UIM runs in the modem, so it is initialized at the same time as MMGSDI and receives all the events generated during the initialization
- QMI UIM sends an indication to all subscribed clients when the status of the card changes
- The card status contains the following:
  - Card state
  - Number of available cards
  - Error cause (only valid in case of a card error state)
  - List of all applications on each card
  - Indication of which application is used for subscriptions
  - Status of each application
    - PIN1 status (or UPIN status) and number of retries
    - PIN2 status and number of retries
    - Perso status and number of retries
    - AID (required to access the application if it is nonprovisioning)
- Client is responsible for comparing the new card status with the previous one to see what changed
- Client can request the card status at any time using a QMI request
- QMI UIM caches the card status, so it can reply quickly without accessing the card

Subscription OK – Personalization on the Client

- During card initialization, the modem performs personalization checks to verify that the SIM card inserted in the device is valid before the subscription is published and the modem starts the procedure to acquire the network.
- The modem can be configured so a client can execute additional personalization checks to augment or replace the current checks. The client can check whether this feature is active during initialization using the QMI_UIM_GET_CONFIGURATION command. When the TLV with the halt_subscription field is missing, the feature is not supported.

Subscription OK – Personalization on the Client (cont.)

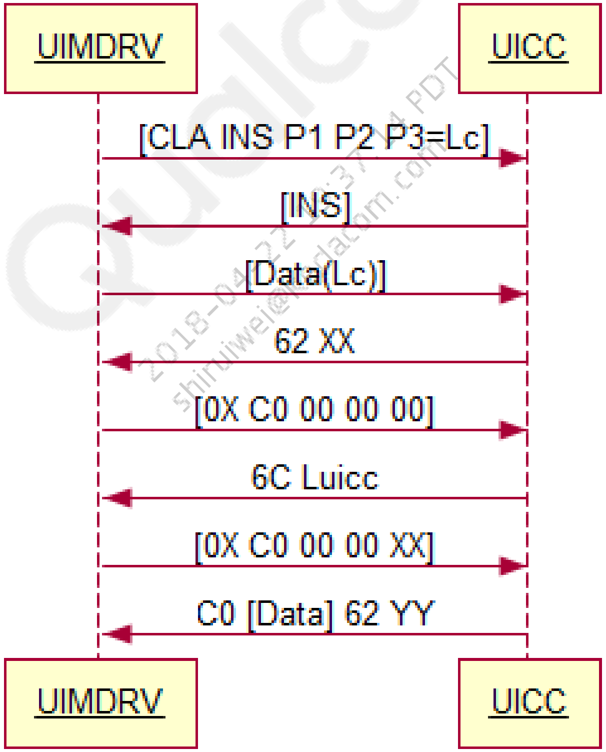

Client
QMI UIM Service
Client checks modem configuration
QMI_UIM_GET_CONFIGURATION_REQ
QMI_UIM_GET_CONFIGURATION_RESP
halt_subscription = TRUE
QMI_UIM_STATUS_CHANGE_IND
app_state = Perso, perso_state = Ready
QMI_UIM_READ_TRANSPARENT_REQ
Session: provisioning, File: IMSI
Additional files can be requested by client, including GID1 and GID2
Client executes the simlock algorithm locally
QMI_UIM_READ_TRANSPARENT_RESP
Result of operation, IMSI value
QMI_UIM_SUBSCRIPTION_OK_REQ
ok_for_subscription = TRUE
QMI_UIM_SUBSCRIPTION_OK_RESP
QMI_UIM_STATUS_CHANGE_IND
app_state = Ready

QMI_UIM_SET_APDU_BEHAVIOR

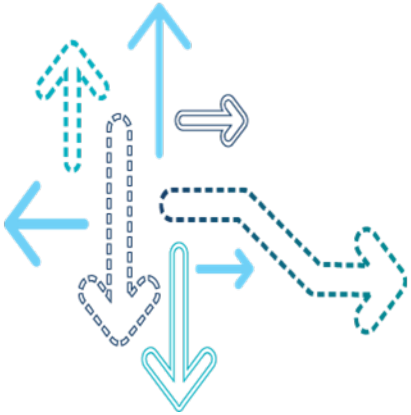
- FR36578 adds a new request/response QMI_UIM_SET_APDU_BEHAVIOR, with channel_id and apdu_behavior to set the behavior of modem for warning SW1-SW2, conforming to the setTransmitBehaviour() API defined by the Open Mobile API Specification version 3.2.
- If transmit behavior is set to FALSE, the modem issues a GET RESPONSE command, according to article “6.1.1 General Rules for Handling of Status Word” of the *Open Mobile API Specification* version 3.2.
- If transmit behavior is set to TRUE:
  - If the UICC returns data and warning SW (62xx or 63xx), the returned data and the warning SW are provided to the calling mobile application and no GET-RESPONSE command is sent.
  - If the UICC returns a warning SW (62xx or 63xx) without data, UIMDRV issues a GET RESPONSE command with Le=00. After sending the GET RESPONSE with Le=00, general rules specified in Chapter 6.1.1 apply for retrieving the data.
    - Modem returns the data together with the first received warning SW
- In the case of openBasicChannel, openLogicalChannel and selectNext(), the GET RESPONSE command is sent if there is a SW warning (62xx or 63xx) independent of the setting in setTransmitBehaviour().
- Modem behavior with the telecom application is not impacted by the API.

QMI_UIM_SET_APDU_BEHAVIOR (cont.)

- Example modem call flow if transmit behavior is set to TRUE, and UICC returns a warning SW (62xx or 63xx) without data.



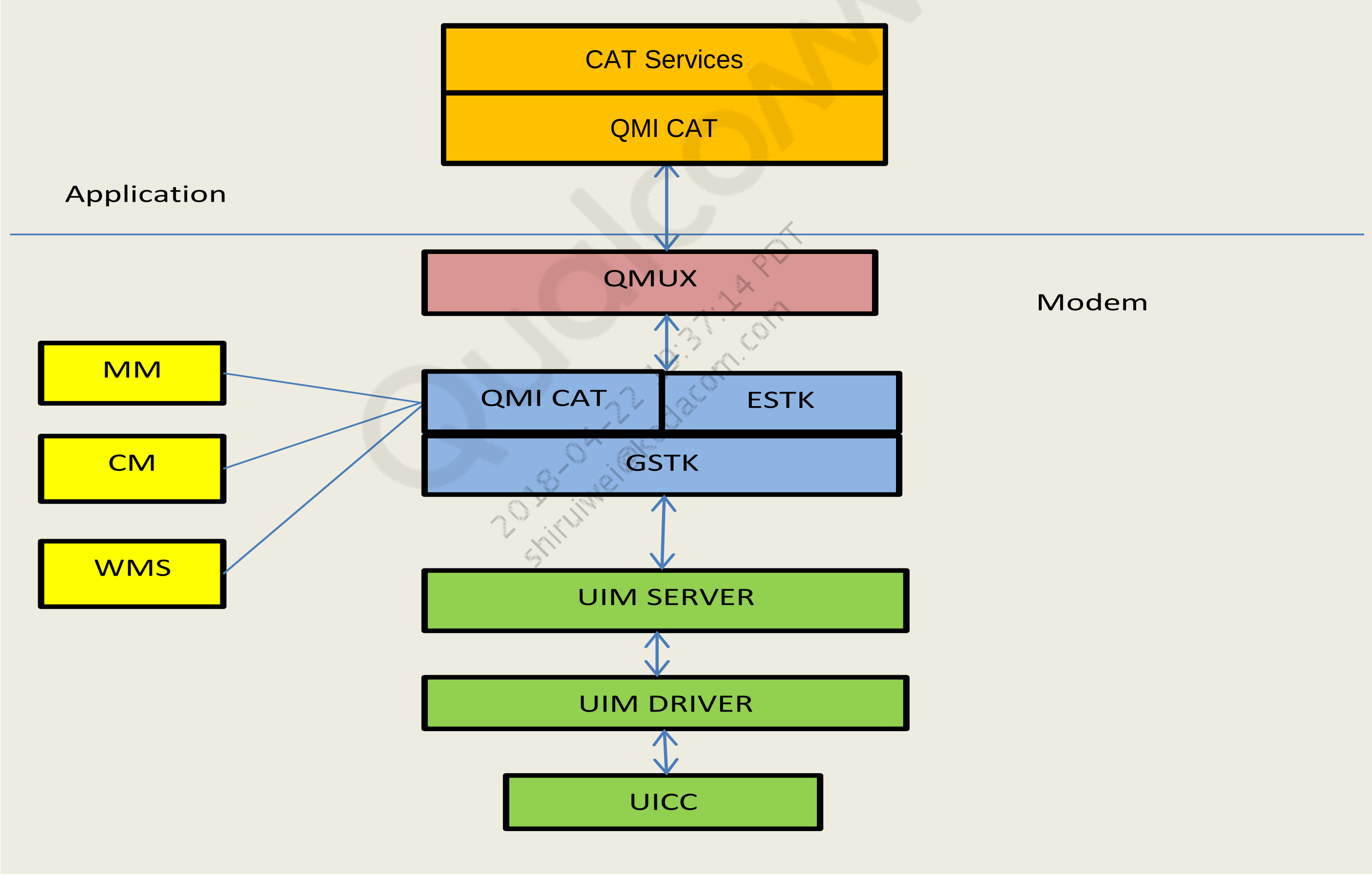
QMI CAT



QMI CAT Description

- QMI for the Card Application Toolkit (QMI CAT) is one of the QMI services
  - Refer to general QMI documentation for more details on the architecture
- It uses the QMUX to communicate with clients
- Service ID is 0x0A
- QMI CAT Specification is available in Agile; refer to *QMI CAT 2.23 for MPSS.DI.2.0, QMI Card Application Toolkit Spec* (80-ND602-11)
- Being adapted and extended to support the High Level Operating System (HLOS)

QMI CAT Architecture



Initialization

- QMI CAT relies on the Enhanced SIM Toolkit (ESTK) module to handle many of the proactive commands
  - Send SMS
  - Send USSD
  - Setup call
  - Bearer Independent Protocol (BIP) commands
- At power-up, QMI CAT sends a Terminal Profile (TP) download command, indicating the commands that the service supports
  - QMI CAT does not know which clients connect later
  - This is a generic request with all UI-related proactive commands
  - With NV item 65683 = 5 or 6, a customizable terminal profile is encoded as in the file /nv/item_files/modem/qmi/cat/qmi_cat_custom_tp.

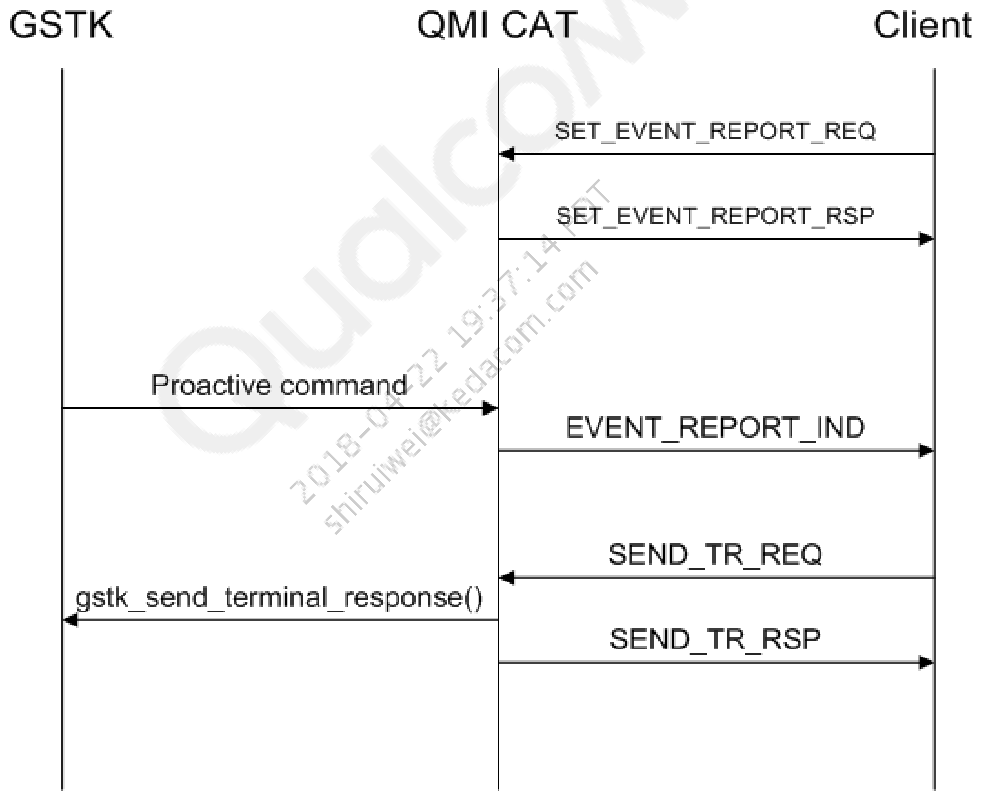
Command Handling

- QMI CAT automatically registers for specific Generic SIM Application Toolkit (GSTK) events at power-up.
  - At this point, QMI CAT does not know which clients are going to use the interface, so it registers for all UI-related proactive commands.
- Each client can register for QMI CAT notifications.
  - Only one client can be registered for each event.
  - If a second client tries to register, it receives an error.
- When a proactive command is received:
  - If a client already exists to receive the command, QMI CAT forwards it to the client. It is the responsibility of the client to send the Terminal Response (TR).
  - If a client does not exist, QMI CAT caches the proactive command.
    - When a client registers for that proactive command, QMI CAT sends it to the client.
    - If no client registers for that proactive command, GSTK times out.

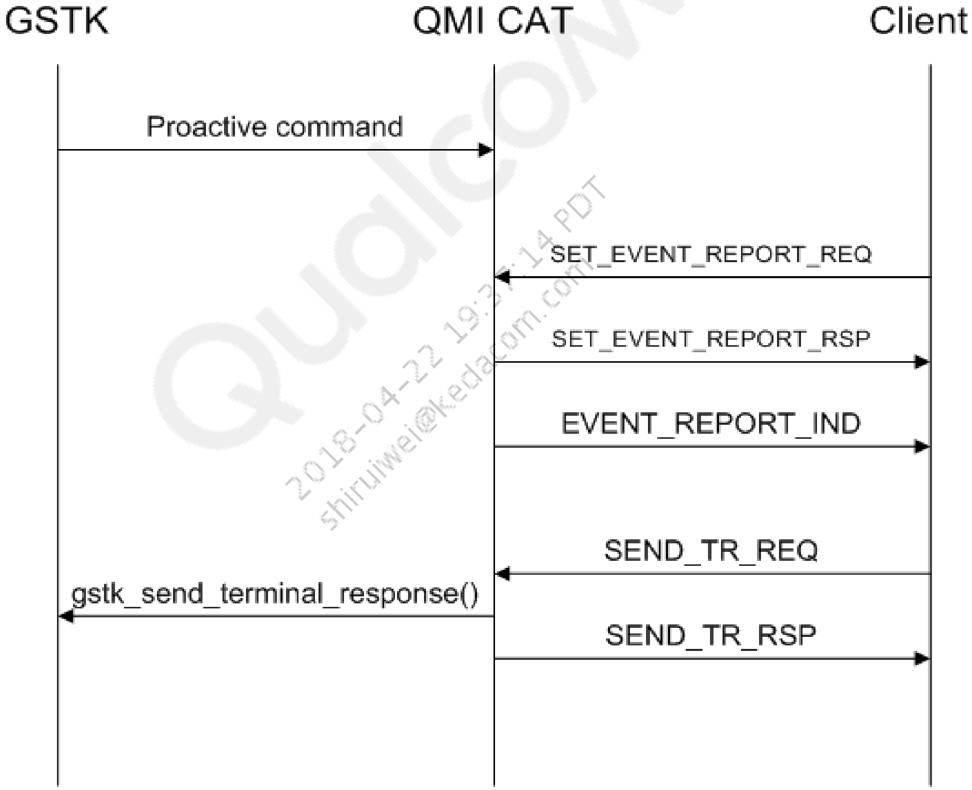
NV Configuration Details

- NV item 65683 – QMI CAT mode
  - The behavior of the QMI CAT interface is controlled using NV item 65683, which is stored on the device in /nv/item_files/modem/qmi/cat/qmi_cat_mode.
  - The file has only 1 byte, with the following possible values:
    - 0 – QMI CAT is disabled
    - 1 – Indications are in RAW format, but only alpha is passed for Send SMS
    - 2 – Indications are in RAW format, with complete messages also passed for network-related commands
    - 3 – Indications are in decoded format
    - 4 – QMI CAT works in decoded format, but indications are not sent to the control point and must be pulled
    - 5 – Indications are in RAW format and allow a customizable terminal profile
    - 6 – Indications are in decoded format and allow a customizable terminal profile
- NV item 67287 – QMI CAT custom terminal profile

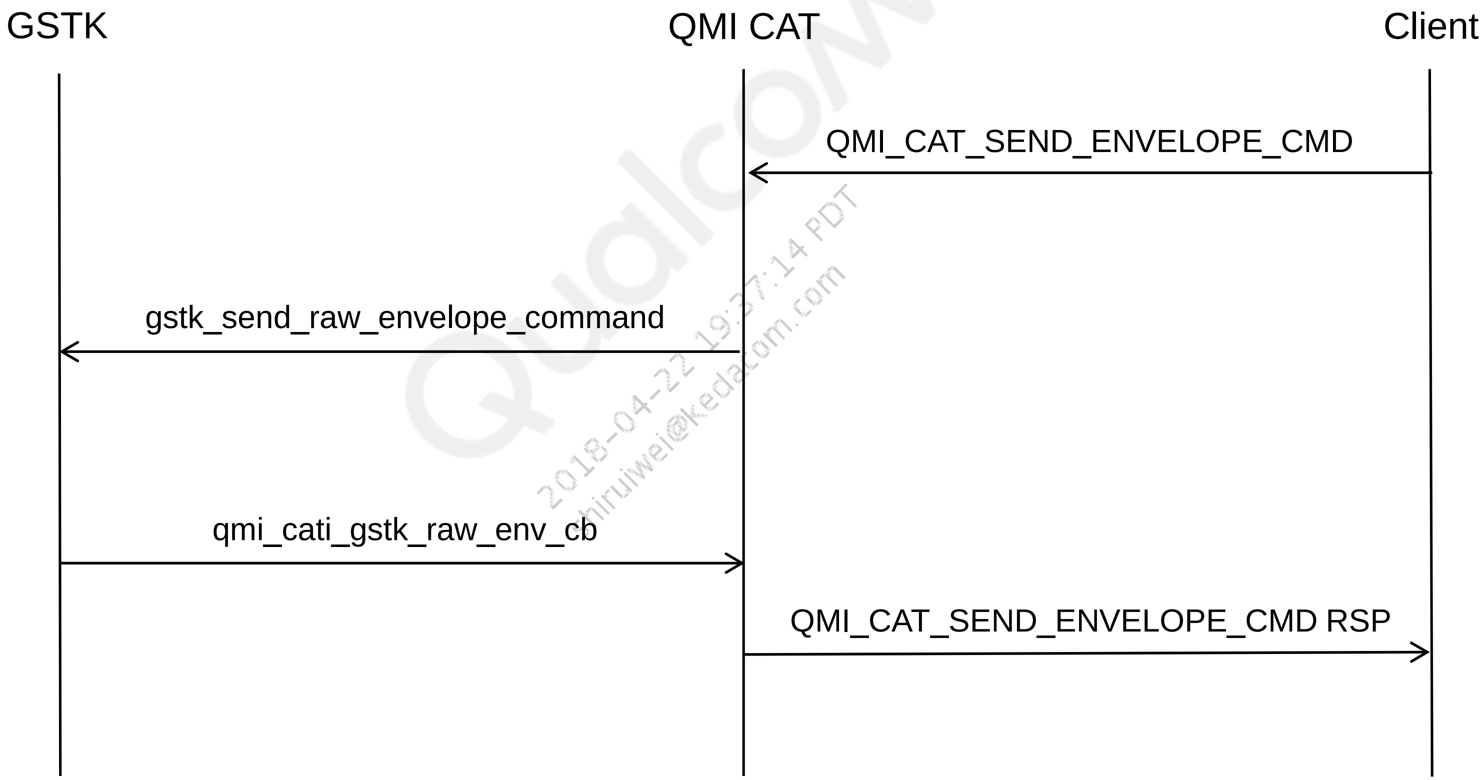
Normal Case – Call Flow



Cache – Call Flow



Envelope – Call Flow

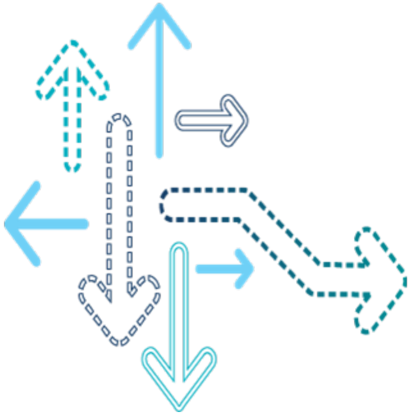


GSTK
QMI CAT
Client
QMI_CAT_SEND_ENVELOPE_CMD
gstk_send_raw_envelope_command
qmi_cati_gstk_raw_env_cb
QMI_CAT_SEND_ENVELOPE_CMD RSP

References

Title	Number
Qualcomm Technologies, Inc.	
QMI CAT 2.23 for MPSS.DI.2.0, QMI Card Application Toolkit Spec	80-ND602-11
Resources	
Open Mobile API Specification	V3.2

Acronym or term	Definition
CAT	Card Application Toolkit
ESTK	enhanced SIM toolkit
GSTK	generic SIM application toolkit
MMFSDI	multimode generic SIM driver interface
QMI	Qualcomm messaging interface
QMUX	QMI Multiplexing Protocol
UIM	user identity module



Questions?

https://createpoint.qti.qualcomm.com