

Explaining the **EMV transaction flow** (from start to end) involves a detailed step-by-step interaction between the **terminal (POS)** and the **smart card (chip)**. This includes APDU commands, EMV tags (from ISO 7816 & EMV 4.3 specs), and cryptographic checks.

Let's break it down in **chronological steps** with what the terminal and card do, and what tags are involved at each step:

EMV Chip Transaction Flow – Step by Step

1. Card Insertion & ATR (Answer to Reset)

- ◆ **Terminal Action:** Resets the chip and waits for a response.
 - ◆ **Card Response:** Sends ATR (Answer To Reset) – indicates supported protocols and features.
 - **Purpose:** Negotiates communication protocol (usually T=0 or T=1).
 - **No EMV tags involved yet.**
-

2. PSE or AID Selection (Application Selection)

- ◆ **Terminal Action:** Sends **SELECT** command to search for applications (using PSE or known AIDs).
- ◆ **Card Response:** Provides application(s) it supports.
 - **Command:** 00 A4 04 00 (Select File)
 - **Tags:**
 - 4F – Application Identifier (AID)
 - 50 – Application Label
 - 9F38 – PDOL (Processing Options Data Object List)

 If PSE (1PAY.SYS.DDF01) is not supported, the terminal uses known AIDs directly.

3. Read Application Data (File Reading)

- ◆ **Terminal Action:** Reads the Application File Locator (AFL) from the response and uses it to **READ RECORD**.
- ◆ **Card Response:** Sends EMV data stored in its files.

- **Command:** 00 B2 XX 0C 00 (Read Record)
 - **Tags:**
 - 57 – Track 2 Equivalent Data
 - 5A – PAN
 - 5F24 – Expiry Date
 - 5F34 – PAN Sequence Number
 - 9F1F – Track 1 Discretionary Data
 - 9F6C – Magstripe Application Version Number
 - 9F10 – Issuer Application Data
-

4. GPO (Get Processing Options)

- ◆ **Terminal Action:** Sends GPO command to determine transaction processing path.
 - ◆ **Card Response:** Returns AIP (what features the card supports) and AFL (files to read).
 - **Command:** 80 A8 00 00
 - **Data:** Built from **PDOL** if available.
 - **Tags:**
 - 9F38 – PDOL
 - 77 or 80 – Response template
 - Inside that:
 - 82 – AIP (Application Interchange Profile)
 - 94 – AFL (Application File Locator)
-

5. Read Records (Based on AFL)

- ◆ **Terminal Action:** Reads records using the AFL.
- ◆ **Card Response:** Sends data used for CVM, risk management, and decision logic.
 - Same **READ RECORD** command
 - Tags from Step 3 + possibly:
 - 8E – CVM List

- 9F0D, 9F0E, 9F0F – Risk Management data
 - 9F42 – Currency Code
 - 9F1A – Terminal Country Code
-

6. Cardholder Verification (CVM Processing)

◆ **Terminal Action:** Determines verification method (PIN, signature, none) using CVM list.

◆ **Card Response:** Verification done locally (offline PIN) or online.

- **Tags:**

- 8E – CVM List
 - 9F34 – CVM Results
 - 9F33 – Terminal Capabilities
-

7. Terminal Risk Management

◆ **Terminal Action:** Performs floor limit check, exception file check, and velocity check.

◆ **No card interaction.**

- **Tags:**

- 9F1B – Terminal Floor Limit
 - 9F35 – Terminal Type
-

8. Terminal Action Analysis (TAA)

◆ **Terminal Action:** Decides action – online, offline, or decline.

◆ **Based On:** AIP, transaction data, issuer rules.

- **Tags:**

- 9F21 – Transaction Time
- 9A – Transaction Date
- 9C – Transaction Type
- 5F2A – Transaction Currency

9. First GAC – Generate Application Cryptogram

- ◆ **Terminal Action:** Sends Generate AC command requesting ARQC, AAC, or TC.
- ◆ **Card Response:** Returns cryptogram.
 - **Command:** 80 AE 80 00 (1st GAC request)
 - **Tags Sent:**
 - 9F02 – Amount
 - 9F03 – Cashback amount
 - 9F1A, 9A, 9C, 9F37, 9F33, 9F35, 95, 9F34, 9F1E, etc.
 - **Response Tags:**
 - 80 or 77 template
 - 9F27 – Cryptogram Information Data
 - 9F36 – Application Transaction Counter (ATC)
 - 9F26 – Application Cryptogram
 - 9F10 – Issuer Application Data

10. Online Authorization (if ARQC generated)

- ◆ **Terminal Action:** Sends transaction to issuer via host.
- ◆ **Issuer Response:** Approves, declines, or requests second AC.
 - Online authorization via host system.
 - Response might include:
 - Issuer Authentication Data (IAD)
 - Scripts for card personalization.

11. 2nd GAC (Generate Application Cryptogram)

- ◆ **Terminal Action:** Sends Generate AC command again to complete transaction.
- ◆ **Card Response:** Sends final cryptogram (TC or AAC).
 - **Command:** 80 AE 00 00 (2nd GAC)

- **Response Tags:**

- Same as 1st GAC, but cryptogram type differs (9F27: 40 for TC, 00 for AAC).
-

12. Transaction Completion

- ◆ **Terminal Action:** Acts on card/issuer decision — print receipt, show result.
 - ◆ **Card:** Retains updated counters and logs.
 - Terminal may log:
 - 9F13, 9F4E, 9F41, 9F07, etc.
-

Summary of Critical Tags by Function

Tag Meaning

- 4F AID (Application Identifier)
- 50 Application Label
- 57 Track 2 Equivalent Data
- 5A PAN
- 82 AIP (Application Interchange Profile)
- 94 AFL (Application File Locator)
- 9F10 Issuer Application Data
- 9F26 Application Cryptogram
- 9F27 Cryptogram Information Data
- 9F36 ATC (Application Transaction Counter)
- 9F02 Authorized Amount
- 8E CVM List
- 9F34 CVM Results



EMV Flow with APDU Commands, TLV Data & Responses

1. ATR (Answer to Reset)

- **Sent by terminal:** [Reset command]
 - **Response from card (ATR Example):**
3B 68 00 00 00 73 C8 40 13 00 90 00
(Specifies T=0 protocol, historical bytes, etc.)
-

2. Application Selection – SELECT AID

- **Command:**
00 A4 04 00 07 A0 00 00 00 03 10 10 00
(Select MasterCard AID)
- **Explanation:**
 - CLA: 00
 - INS: A4 (SELECT)
 - P1/P2: 04 00 (select by name)
 - Lc: 07 (length of AID)
 - Data: A0 00 00 00 03 10 10 (MasterCard AID)
- **Response (TLV):**

r

CopyEdit

6F 1A

84 07 A0 00 00 00 03 10 10

A5 0F

50 0B 4D 61 73 74 65 72 43 61 72 64

90 00

- 6F – FCI Template

- 84 – DF Name (AID)
 - 50 – Application Label (e.g., "MasterCard")
-

3. GET PROCESSING OPTIONS (GPO)

- **Command:**
80 A8 00 00 0C 83 0A 00 00 00 00 00 00 00 00 00 00 00

- **Explanation:**

- CLA: 80 (proprietary)
- INS: A8
- Data: 83 0A ... → PDOL data

- **Response (GPO Response – Format 1):**

CopyEdit

80 06

82 02 38 00

94 02 08 01

90 00

- 82 – AIP
 - 94 – AFL (tells which records to read)
-

4. READ RECORD (Based on AFL)

- **Command Example:**
00 B2 01 0C 00 (SFI=01, record=01)
- **Response:**

CopyEdit

70 1A

5A 08 4761XXXXXXXXXXXX

5F24 03 241231

57 13 4761XXXXXXXXXXXXD241231...

90 00

- 70 – Record template
 - 5A – PAN
 - 5F24 – Expiry Date
 - 57 – Track 2 Equivalent
-

5. Cardholder Verification (CVM List)

- CVM is usually found in a previous **READ RECORD** response:

mathematica

CopyEdit

8E 14 00 00 00 00 00 01 01 02 02 03 03 04 04 05 05 06 06

- 8E – CVM List
 - Terminal uses this list to verify PIN/signature based on capability.
-

6. Generate AC – First (GAC 1)

- **Command:**

80 AE 80 00 XX [CDOL1 Data]

- **CDOL1 Includes Tags:**

9F02, 9F03, 9F1A, 95, 5F2A, 9A, 9C, 9F37, etc.

Example CDOL1 Data:

scss

CopyEdit

00 00 00 01 00 00 (Amount)

00 00 00 00 00 00 (Cashback)

02 50 (Terminal Country Code)

... etc.

- **Response:**

mathematica

CopyEdit

77 28

9F27 01 80

9F36 02 00 0C

9F26 08 7A 3F D5 6B ...

9F10 0E 06 01 1A A0 ...

90 00

- 9F27 – CID (Cryptogram Info Data) → 80 means ARQC
 - 9F26 – Application Cryptogram (ARQC)
 - 9F36 – ATC (Application Transaction Counter)
-

7. Online Authorization (*Not APDU, but host communication*)

- Terminal sends:
 - ARQC (9F26)
 - PAN (5A)
 - Track data (57)
 - IAD (9F10)
 - ATC (9F36)
 - Transaction data
 - Host replies with:
 - Response code (8A)
 - Authorization Response Cryptogram (ARPC)
 - Issuer Scripts (optional)
 - 91 – Issuer Authentication Data
-

8. Generate AC – Second (GAC 2)

- **Command:**
80 AE 00 00 XX [CDOL2 Data]
- **Response:**

mathematica

CopyEdit

77 28

9F27 01 40

9F36 02 00 0D

9F26 08 A1 B2 C3 D4 ...

9F10 0E 06 01 1A A0 ...

90 00

- 9F27 = 40 (indicates TC = Transaction Certificate)
 - Transaction is approved and can be finalized.
-

9. Issuer Scripts (if any)

- **Command:**
80 86 00 00 [Script Data]
 - Used to update card state, counters, etc.
-

Final Tags Summary for Developers

Step	APDU Command Key Tags Used/Returned	
Select AID	00 A4 04 00	4F, 50, 84
GPO	80 A8 00 00	82, 94
Read Record	00 B2 XX 0C 00	5A, 57, 5F24, 8E, 9F10
Generate AC 1	80 AE 80 00	9F26, 9F27, 9F36, 9F10, 9F34
Generate AC 2	80 AE 00 00	9F26, 9F27, 9F36, 9F10

1. ATR (Answer to Reset) – In-Depth Explanation

What is ATR?

ATR (Answer to Reset) is a response sent by a smart card when it is inserted into a terminal and **power is applied**. It's the card's way of saying:

“Hi, I'm ready. Here's how you can talk to me.”

Purpose of ATR

- It **initializes the communication** between the **card** and the **reader (terminal)**.
 - Tells the terminal:
 - What **protocols** the card supports (T=0, T=1)
 - Whether it uses **synchronous or asynchronous** communication
 - What **clock rate** and **timing** it requires
 - Any **historical bytes** (card-specific info)
-

How ATR Works

When a card is inserted:

1. The terminal applies power.
 2. The card's chip resets internally.
 3. The card sends an ATR byte stream over the serial line.
-

Structure of ATR (ISO/IEC 7816-3 standard)

The ATR is a **variable-length** string of bytes (typically 20–30 bytes). It's structured like this:

graphql

CopyEdit

TS | T0 | Interface Bytes (TAi, TBi, TCi, TDi) | Historical Bytes | TCK

1. TS (1 byte) – Initial character

- Indicates direct (TS = 3B) or inverse (TS = 3F) convention
Example: 3B = direct convention

2. T0 (1 byte) – Format character

- High nibble: Number of historical bytes (H)
- Low nibble: Presence of interface bytes TA1, TB1, TC1, TD1

3. Interface Bytes (TAi, TBi, TCi, TDi)

Define parameters like:

- **BAUD rate**
- **Clock frequency factor (Fi)**
- **Bit rate adjustment (Di)**
- **Voltage class**
- **Supported protocols (T=0, T=1)**

Each TD indicates whether **another group of interface bytes (TA2, TB2, etc.)** follows.

4. Historical Bytes (up to 15)

Card-specific information, such as:

- Card type
- Manufacturer
- Application support (e.g., Visa, MasterCard, etc.)

5. TCK (1 byte) – Check byte

- Present only if protocol is **T ≠ 0** (i.e., T=1 or more).
- XOR checksum of all bytes from T0 to the last interface byte.

Example ATR Breakdown

Example ATR:

CopyEdit

3B 65 00 00 00 73 C8 40 13 00 90 00

Breakdown:

Byte	Meaning
3B	TS – Direct Convention
65	T0 – 5 historical bytes (H=5), TD1 present 00 00 00 TA1, TB1, TC1 (all 0)
73	TD1 – Protocol T=1, more interface bytes follow
C8	TA2 – Specific protocol (e.g., T=1 forced)
40	TB2
13	TC2
00 90 00	Historical bytes or trailer (custom usage)

⚠ Note: Real-world ATRs may differ based on card issuer, chip type, and supported features.

❖ How Terminal Uses ATR

Once ATR is received:

- The terminal **parses** it.
 - Configures communication:
 - **Bit rate**
 - **Protocol (T=0 or T=1)**
 - **Waiting time**
 - Moves to the **Application Selection** step (SELECT AID)
-

🔑 Summary

ATR Element	Meaning
TS	Convention (Direct/Inverse)
T0	Format + Historical Bytes Count
TAx-TDx	Communication parameters

ATR Element Meaning

Historical Bytes Card profile/application hints

TCK Checksum for T≠0

Would you like me to decode an actual **ATR from a real EMV card** or simulate it in a POS log format for training/testing?

You said:

what is T0 and T1 and how they get selected

ChatGPT said:

Great question! Let's break it down clearly:

What Are T=0 and T=1 in Smart Cards?

These refer to **communication protocols** between a **smart card (like EMV chip)** and a **card reader (POS terminal)** — defined by **ISO/IEC 7816-3**.

T=0 Protocol (Character-level Transmission)

- **Oldest and most widely supported**
- **Byte-oriented**, sends one byte at a time.
- Each byte is acknowledged before sending the next.
- Has **simpler logic**, less overhead.

◆ **Used for:**

- Many **EMV contact cards**
- SIM cards
- Simpler POS devices

Characteristics:

- Half-duplex (one-way at a time)
- Simpler state machine
- Minimal RAM required on card

T=1 Protocol (Block-level Transmission)

- **Block-oriented** communication
 - Transmits data in blocks with **error correction** (EDC)
 - Supports **more complex data exchange**
 - Includes mechanisms for **retransmission** and **flow control**
- ◆ **Used for:**
- **Newer EMV cards**
 - Smart cards with **secure elements**
 - High-performance terminals

Characteristics:

- Allows larger APDU responses
- Supports **error checking** with checksums (TCK)
- More efficient for complex interactions

How T=0 or T=1 Is Selected?

1. Card Declares Its Supported Protocols in ATR

The card includes a **TDx byte** in the **ATR (Answer To Reset)** to tell the terminal which protocols it supports.

Example:

In ATR:

CopyEdit

3B 65 00 00 00 73 C8 40 13 00 90 00

The TD1 byte = 73

- High nibble (7): indicates further interface bytes
- Low nibble (3): tells that **T=1** is supported

► A card can support **both T=0 and T=1**, or just one.

2. Terminal Chooses Protocol Based on ATR

The terminal:

- Reads **TD1**, **TD2**, etc. to see supported protocol(s)
 - Chooses a compatible one
 - Sends a **PTS (Protocol and Parameters Selection)** request if it wants to switch
-

Example PTS (Protocol Type Selection)

If card supports both T=0 and T=1, the terminal may send PTS to switch:

- **PTS Request:** FF 10 95 00
 - If card agrees, it responds with PTS response
 - If PTS exchange fails, default protocol (usually T=0) is used
-

In Summary

Feature	T=0	T=1
Type	Byte-oriented	Block-oriented
Error Handling	No error detection	Supports error detection (LRC/CRC)
Speed	Slower	Faster (with flow control)
Memory Use	Low	Higher (requires buffers)
ATR TDi Byte	T=0 = 00, T=1 = 01	T=1 = 01 in lower nibble
Typical Use	EMV, SIM, old terminals	Newer EMV, Secure apps

Let's go **in-depth** into **Step 2: Application Selection – SELECT AID**, a crucial part of the **EMV flow**. This is where the terminal asks the card what application to use (e.g., Visa, MasterCard, RuPay) for the transaction.

2. Application Selection – SELECT AID (Detailed Explanation)

Goal of This Step

- Identify which **EMV payment application** (AID) on the card should be used.
 - Each application (Visa, MasterCard, RuPay, etc.) has its own **AID (Application Identifier)**.
-

How It Works

1 Option 1: PSE (Payment System Environment) – Indirect Selection

- Terminal first selects **1PAY.SYS.DDF01** (PSE) to discover supported applications.
- If the card supports PSE, it returns a list of available AIDs.

Command:

00 A4 04 00 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31
(ASCII = 1PAY.SYS.DDF01)

Response:

TLV with a list of SFI + AIDs

Card might say “**File not found**” (**6A82**) if it doesn’t support PSE. Then terminal tries Option 2.

2 Option 2: Direct AID Selection

Terminal uses **pre-configured list of known AIDs** (Visa, MasterCard, RuPay, etc.) and tries them one by one.

◆ **SELECT AID APDU Command Format:**

less

CopyEdit

CLA | INS | P1 | P2 | Lc | Data (AID) | Le

◆ **Example – Select MasterCard AID:**

- **Command:**

00 A4 04 00 07 A0 00 00 00 04 10 10

- CLA: 00
- INS: A4 (SELECT)
- P1/P2: 04 00 (select by name)
- Lc: 07 (length of AID)
- Data: AID for MasterCard A0 00 00 00 04 10 10

AID is defined by **ISO/IEC 7816-5** and assigned by payment schemes.



Card Response – SELECT Response (FCI Template)

The card responds with File Control Information (FCI):

◆ **Example Response:**

mathematica

CopyEdit

6F 1E

84 07 A0 00 00 00 04 10 10

A5 13

50 0B 4D 41 53 54 45 52 43 41 52 44

9F38 06 9F66 9F02 9F03 9F1A 95 5F2A

90 00

◆ **Explanation:**

- 6F – FCI Template (root structure)
- 84 – DF Name (AID) returned by card
- A5 – Proprietary data (subfields)
 - 50 – Application Label ("MASTERCARD")
 - 9F38 – PDOL (Processing Options Data Object List)

What the Terminal Does Next

1. Parses the FCI response.
 2. Stores PDOL if present.
 3. Displays Application Label to the user (optional).
 4. Moves to **GPO (Get Processing Options)**.
-

Common AIDs (Examples)

Scheme	AID
Visa	A0 00 00 00 03 10 10
MasterCard	A0 00 00 00 04 10 10
RuPay	A0 00 00 05 24 10 10
Amex	A0 00 00 00 25 01
Discover	A0 00 00 01 52 30 10

Error Responses

SW1 SW2 Meaning

6A82	File not found (AID not supported)
6A84	Not enough memory
6985	Conditions not satisfied

If card returns 6A82, the terminal tries next AID in its list.

Developer Tip: Use Logs Like

yaml

CopyEdit

> APDU: 00 A4 04 00 07 A0 00 00 00 03 10 10

< Response: 6F XX ... 90 00

To debug app selection issues.

 **Summary Table**

Field Value Meaning

CLA 00 ISO Command

INS A4 SELECT

P1 04 Select by name

P2 00 First or only occurrence

84 Tag DF Name (returned AID)

50 Tag Application Label

9F38 Tag PDOL (for GPO step)

Absolutely! Let's dive **in-depth** into **Step 3: GET PROCESSING OPTIONS (GPO)**—a critical phase of the EMV flow that defines how the terminal and card will proceed with the transaction.

3. GET PROCESSING OPTIONS (GPO) – In-Depth Explanation

Purpose of GPO

After the terminal selects an application (AID), it needs to know:

- What the card supports (offline/online auth, CVM methods, etc.)
- Which records to read (via AFL – Application File Locator)

This is done by sending a **GET PROCESSING OPTIONS (GPO)** command to the card.

APDU Command Format

less

CopyEdit

CLA | INS | P1 | P2 | Lc | Data Field | Le

80 | A8 | 00 | 00 | XX | 83 XX [PDOL Data] | 00

- CLA: 80 – Proprietary class
- INS: A8 – GET PROCESSING OPTIONS
- P1/P2: 00 00
- Lc: Length of data field
- Data Field: Constructed Data Object List (tag 83) that contains **PDOL-related values**
- Le: 00 (maximum expected length)

Example GPO Command

mathematica

CopyEdit

80 A8 00 00 0C 83 0A 00 00 00 00 00 00 00 00 00 00 00

Explanation:

- 0C: total data length
- 83 0A: tag 83 (length 0A = 10 bytes)
- 00 00 00 00 00 00 00 00 00 00: dummy values for PDOL

The **PDOL** is obtained from the 9F38 tag in the SELECT response.

What is PDOL?

PDOL = Processing Options Data Object List

- List of tags (and lengths) that the **card wants** the terminal to send in GPO
- Defined by the card, parsed by the terminal.

Example PDOL (9F38):

CopyEdit

9F66 04 9F02 06 9F03 06 9F1A 02 95 05 5F2A 02

- Terminal must build 83 template with the values for these tags, in order.
-

Card Response to GPO

There are **two response formats** (EMV 4.3):

Format 1 (tag 80) — Most common

mathematica

CopyEdit

80 XX

82 XX XX ; AIP (Application Interchange Profile)

94 XX... ; AFL (Application File Locator)

90 00

Example:

CopyEdit

80 06

82 00 38 ; AIP
94 10 01 02 02 03 04 01
90 00

Format 2 (tag 77) — TLV wrapped format

CopyEdit

77 XX

82 XX XX ; AIP
94 XX... ; AFL

90 00

Meaning of Key Tags

AIP (Application Interchange Profile) – 82

2 bytes defining what the card supports:

Bits (Byte 1) Feature

- b8 = 1 Offline static data authentication
- b7 = 1 Offline dynamic data authentication
- b6 = 1 Cardholder verification supported
- b5 = 1 Terminal risk management
- b4 = 1 Issuer authentication supported
- b3 = 1 Terminal/certificate data authentication
- b1 = 1 CDA supported (only in Byte 2)

AFL (Application File Locator) – 94

- Tells terminal **which SFI + record ranges** to read

Each AFL entry = **4 bytes**:

- Byte 1: SFI (5 bits) + logical structure
- Byte 2: Start record

- Byte 3: End record
- Byte 4: Number of records with SDAD

► Terminal parses AFL and issues **READ RECORD** commands next.

Common GPO Errors

SW1 SW2 Meaning

6A80	Incorrect data (PDOL length mismatch)
6985	Conditions of use not satisfied
6700	Wrong length (bad Lc or Le)

Terminal Logic in GPO Step

1. Parses PDOL from 9F38 tag
 2. Constructs data for 83 using terminal environment
 3. Sends GPO
 4. Parses 82 (AIP) and 94 (AFL)
 5. Prepares to read records using AFL
-

Summary

Item	Tag	Purpose
AIP	82	Tells terminal what features the card supports
AFL	94	Tells terminal which records to read next
PDOL	9F38	Tells terminal what data to send in GPO
GPO Command	80 A8	Core instruction to initialize EMV processing

Absolutely! Let's go **in-depth** into **Step 4: READ RECORD (Based on AFL)** – this is where the terminal collects data stored in the EMV chip, based on what the card told it during the GPO step.

4. **READ RECORD (Based on AFL) – In-Depth**

Purpose

After the card responds to the GPO command with the **AFL (Application File Locator)**, the terminal uses it to know:

- **Which files (SFI)** to read
- **Which records** within those files
- **In what order**

The **READ RECORD** command is used to retrieve these records from the card.

What is AFL (Tag 94)?

AFL is a list that tells the terminal which records it can request from the card.

Each AFL entry is **4 bytes**:

mathematica

CopyEdit

Byte 1: SFI (upper 5 bits), always left-shifted by 3

Byte 2: First record number

Byte 3: Last record number

Byte 4: Number of records for SDA (Static Data Authentication)

Example AFL (94 tag in GPO response)

CopyEdit

94 10 01 02 02 03 04 01

Breaks down to two AFL entries:

Entry 1:

- **SFI:** 2 ($10 \gg 3 = 2$)
- **First record:** 1
- **Last record:** 2
 - Read SFI 2, Records 1 and 2

Entry 2:

- **SFI:** 3 ($04 \gg 3 = 3$)
- **First record:** 3
- **Last record:** 4
 - Read SFI 3, Records 3 and 4

READ RECORD – APDU Command

◆ Command Format:

mathematica

CopyEdit

CLA | INS | P1 | P2 | Le

00 | B2 | Record Number | (SFI << 3 | 4) | 00

- CLA: 00 (standard class)
- INS: B2 (READ RECORD)
- P1: Record number (01, 02, etc.)
- P2: SFI << 3 + 4 (logical OR with 04 as fixed bits)
- Le: 00 (expect max response)

◆ Example Command

Reading Record 1 from SFI 2:

mathematica

CopyEdit

00 B2 01 0C 00

- 0C = (SFI 2 << 3) | 4 = 16 + 4 = 0C

 **Card Response: TLV Data**

Response contains EMV data used in the transaction:

◆ **Example:**

mathematica

CopyEdit

70 1E

5A 08 4761XXXXXXXXXXXX

5F24 03 241231

5F34 01 01

9F1F 18 313233...

90 00

 **Common Tags in Records:**

Tag Description

5A Application Primary Account Number (PAN)

5F24 Application Expiry Date

5F34 PAN Sequence Number

57 Track 2 Equivalent Data

8E CVM List

9F10 Issuer Application Data

9F1F Track 1 Discretionary Data

9F36 Application Transaction Counter (ATC)

9F07 Application Usage Control

 **Terminal Logic:**

1. Parses AFL from GPO response.

2. Loops through each entry:
 - o For each SFI and record range:
 - Issues a **READ RECORD** APDU.
 - Stores or parses the returned TLV data.
 3. Uses this data in:
 - o Cardholder Verification (CVM)
 - o Offline authentication
 - o Cryptogram generation (GAC)
 - o Building CDOL (Card Data Object List)
-

Error Handling

SW1 SW2 Meaning

6A83	Record not found (wrong SFI/record)
6A82	File not found (SFI doesn't exist)
6985	Conditions of use not satisfied

Summary

APDU Purpose READ RECORD

Command Header 00 B2

Record Number 01, 02, etc.

SFI (<<3) OR

Response Format TLV (starts with 70)

Tags Returned 5A, 57, 8E, 9F10, etc.

Follows GPO Yes, directly after

Great! Let's dive in-depth into **Step 5: Cardholder Verification (CVM List)** — a key part of the EMV transaction process that ensures the cardholder is legitimate.

5. Cardholder Verification (CVM List) – In-Depth

Goal of Cardholder Verification

To confirm that the person presenting the card is the **actual cardholder**.

This step decides:

- Should the user **enter a PIN**?
- Is **signature required**?
- Or should **no verification** be done (e.g., low-value contactless)?

The **CVM List (tag 8E)** defines the **rules and priority** for verifying the cardholder.

Where is the CVM List Found?

- Sent from the card in one of the **READ RECORD** responses after the **GPO** step.
 - TLV tag: 8E
-

Example:

mathematica

CopyEdit

8E 10

01 01 02 02 03 03 04 04 05 05 1E 03 1F 00

- 8E – CVM List
 - 10 – Length (16 bytes)
 - Followed by **pairs of bytes** (each 2-byte entry is a CVM rule)
-

CVM List Structure

- Format:

X1 X2 = CVM Rule

- X1: CVM Code (what to do)
- X2: Condition Code (when to apply)

Byte 1 (CVM Code) Meaning

01	Plaintext PIN verified by ICC
02	Encrypted PIN verified online
03	Plaintext PIN + Signature
1E	Signature (paper)
1F	No CVM required
04, 05, etc.	Reserved/other CVM methods

Byte 2 (Condition Code)	When to apply
00	Always
01	If unattended cash
02	If not unattended cash
03	If terminal supports PIN
04	If manual cash
05	If purchase
1F	Always (used with no CVM)

How Terminal Uses the CVM List

1. Reads tag 8E from the card.
2. Iterates through CVM rules **in order**.
3. Evaluates each condition based on:
 - Terminal type
 - Transaction type (purchase, refund, etc.)

- Capabilities (does it support PIN entry?)
4. When it finds a matching rule:
- Performs the CVM (e.g., PIN entry, signature prompt).
5. Stores result in tag 9F34 (CVM Results).
-

Tag 9F34 – CVM Results

- Sent later in **Generate AC (GAC)** command.
- Format: CVM Code | Condition Code | CVM Result
- Example:

CopyEdit

9F34 03 42 03 01

- 42: CVM performed = Online encrypted PIN
 - 03: Condition matched
 - 01: CVM successful
-

Sample CVM List and Meaning

pgsql

CopyEdit

8E 0E

02 03 → Encrypted PIN if terminal supports it

1E 05 → Signature if purchase

1F 00 → No CVM always

- Try encrypted PIN (if terminal supports)
 - Else ask for signature (if purchase)
 - Else skip CVM
-

CVM Decision Factors

Terminal Property Relevant Tags

CVM Capabilities 9F33

Terminal Type 9F35

Transaction Type 9C

Terminal Country 9F1A

CVM Failure Handling

If the selected CVM method fails (e.g., wrong PIN), the terminal may:

- Try fallback CVM from the list
 - Go online (if required)
 - Reject the transaction (based on policy)
-

Summary Table

Item	Tag / Field Description	
CVM List	8E	Rules to verify cardholder
CVM Result	9F34	Outcome of applied CVM
CVM Capabilities	9F33	Terminal's ability (PIN, signature)
Transaction Type	9C	Tells if purchase, cash, refund, etc.

Absolutely! Let's now go **deep into Step 6: Generate Application Cryptogram (GAC 1)** — one of the most crucial stages in an EMV transaction, where the terminal asks the card to produce a **cryptographic response** that determines whether the transaction should proceed **offline, online, or be declined**.

6. Generate AC – First (GAC 1) – In-Depth

Purpose

The first **Generate AC (Application Cryptogram)** command is issued by the **terminal** after:

1. Reading necessary data using **READ RECORD**
2. Performing **CVM (Cardholder Verification)**
3. Running **Terminal Risk Management**

 The terminal now **asks the card to generate a cryptogram**, which helps determine:

- Offline approval or decline (AAC or TC)
- Online processing required (ARQC)

Types of Cryptograms the Card Can Generate

Cryptogram Meaning

ARQC Authorization Request Cryptogram (go online)

TC Transaction Certificate (approve offline)

AAC Application Authentication Cryptogram (decline offline)

GAC 1 – APDU Command Format

less

CopyEdit

CLA | INS | P1 | P2 | Lc | CDOL1 Data | Le

80 | AE | 80 | 00 | XX | [CDOL1] | 00

- CLA: 80 – Proprietary class
 - INS: AE – GENERATE AC
 - P1: 80 – Request **ARQC**
 - P2: 00 – Not used
 - Lc: Length of CDOL1 data
 - Data: Terminal-prepared data elements from **CDOL1**
 - Le: 00 – Max expected response length
-

What is **CDOL1**?

CDOL1 = Card Data Object List 1

- List of EMV tags (with lengths) the **card expects** in GAC 1.
 - Defined by the card in one of the **READ RECORD** responses.
 - Terminal reads CDOL1 and prepares matching data.
-

Example **CDOL1 (From card's record)**:

pgsql

CopyEdit

8C 1A

9F02 06 → Amount Authorized

9F03 06 → Amount Other

9F1A 02 → Terminal Country Code

95 05 → Terminal Verification Results

5F2A 02 → Transaction Currency Code

9A 03 → Transaction Date

9C 01 → Transaction Type

9F37 04 → Unpredictable Number

→ Terminal fills these tags into a **single byte stream**, in order.

Sample GAC 1 Command

mathematica

CopyEdit

80 AE 80 00 1A

00 00 00 01 00 00 → Amount

00 00 00 00 00 00 → Cashback

08 26 → Country Code

80 00 00 00 00 → TVR

08 26 → Currency Code

24 08 14 → Date (YYMMDD)

00 → Transaction Type (Purchase)

6F 2A 3B 4C → Unpredictable Number

00

GAC Response (ARQC, AAC, or TC)

Format 1: 80 Response Template (Simple)

mathematica

CopyEdit

80 XX

9F27 01 80 ; Cryptogram Info Data

9F36 02 00 12 ; ATC

9F26 08 AA BB ... ; Application Cryptogram

9F10 XX ... ; Issuer Application Data

90 00

Format 2: 77 Response Template (TLV)

CSS

CopyEdit

77 XX

9F27 01 80

9F36 02 00 12

9F26 08 AA BB CC DD ...

9F10 0E 06 01 1A A0 B8 ...

90 00

🔑 Key Tags in GAC 1 Response

Tag Meaning

9F27 Cryptogram Information Data (CID)

9F26 Application Cryptogram (ARQC/AAC/TC)

9F36 Application Transaction Counter (ATC)

9F10 Issuer Application Data

9F37 Unpredictable Number (matches request)

9F1A Terminal Country Code (repeated for verification)

🔍 How Terminal Interprets CID (9F27)

Value Meaning

80 ARQC → Send to issuer (online)

40 TC → Offline approval

00 AAC → Offline decline

🧠 Terminal's Next Action Depends on CID:

CID (9F27) Terminal Action

80 (ARQC) Go online with host

40 (TC) Approve and print receipt

CID (9F27) Terminal Action

00 (AAC) Decline and reject txn



Common Errors

SW1 SW2 Meaning

6985 Conditions of use not satisfied (e.g., missing CVM)

6A80 Incorrect data (CDOL mismatch)

6700 Wrong length (Lc mismatch with CDOL1)



Summary

Item Description

GAC 1 First Generate AC command from terminal

APDU Code 80 AE 80 00

Input Data CDOL1 data

Output Tags 9F26, 9F27, 9F36, 9F10, etc.

Output Type ARQC, AAC, or TC

Excellent! Let's go **in-depth** into **Step 7: Online Authorization**, a **critical host-side phase** of the EMV transaction. While this **is not an APDU exchange**, it's essential for understanding how EMV-compliant terminals communicate with the **payment host or issuer bank** to approve or decline a transaction.

7. **Online Authorization – In Depth**

Purpose

The **card** has generated an **ARQC** (Authorization Request Cryptogram), signaling that the **transaction must go online**.

Now, the **terminal prepares and sends a request** to the **issuer bank's host system**, typically via the **acquirer**, to:

- Authenticate the card
 - Authorize the transaction amount
 - Possibly send **Issuer Scripts** back to the card
-

How Online Authorization Works (Step-by-Step)

1. Host Message Preparation

Terminal assembles a host request using the **ARQC**, **card data**, and **transaction data**.

Most commonly, the data is packed in **ISO 8583** format.

Key Fields for Authorization:

ISO Field EMV Tag Description

2	5A, 57	PAN / Track 2 Equivalent
3	—	Processing code (transaction type)
4	9F02	Amount Authorized
11	—	STAN (System Trace Audit Number)
14	5F24	Expiry Date

ISO Field EMV Tag Description

22	—	POS Entry Mode
35	57	Track 2 Data
52	—	PIN block (if PIN entered)
55	—	EMV Data (TLV data including ARQC)

✓ 2. Tag 55 – EMV Data in Authorization Request

Field 55 (DE55) includes **raw TLV EMV tags**, such as:

EMV Tag Description

9F26 Application Cryptogram (ARQC)

9F10 Issuer Application Data

9F36 Application Transaction Counter

9F37 Unpredictable Number

9F02 Authorized Amount

5F2A Transaction Currency

9A Transaction Date

9C Transaction Type

82 AIP

95 Terminal Verification Results

9F1A Terminal Country Code

9F03 Amount Other (e.g., cashback)

9F33 Terminal Capabilities

9F34 CVM Results

9F35 Terminal Type

9F09 Application Version Number

 This field can be 100–300+ bytes long and is **host-parsed** based on EMV rules.

3. Sending the Online Request

The terminal:

- Encrypts sensitive data (e.g., PIN block)
 - Sends the **ISO 8583 message** via:
 - Local payment switch
 - Acquirer
 - To the issuing bank (Visa, MasterCard, etc.)
-

4. Issuer Host Processing

Issuer will:

1. Validate ARQC cryptographically (using keys shared during personalization)
 2. Check card status (hot list, stolen, expired)
 3. Check account balance, limits
 4. Optionally send:
 - **ARPC** (Authorization Response Cryptogram)
 - **Issuer Scripts** (for card updates)
-

5. Response to Terminal

Host replies with:

- Response Code (e.g., 00 = Approved, 05 = Declined)
 - Optional:
 - **ARPC** (Tag 91)
 - **Issuer Authentication Data (Tag 91)**
 - **Issuer Script (Tag 71 or 72)**
-

What Is ARPC?

ARPC = Authorization Response Cryptogram

- Generated by issuer from the original ARQC and other transaction elements
 - Proves to the card that the response is **authentic**
 - Returned as Tag 91 or embedded in Tag 8A or Tag 9F10
-

Tags Commonly Received in Host Response

Tag Meaning

8A Authorization Response Code (e.g., 00)

91 Issuer Authentication Data (ARPC)

71, 72 Issuer Script Templates 1 & 2 (optional commands to execute on card)

89 Script Results (from card to host)

Terminal Behavior Based on Host Response

Host Reply Terminal Does

8A = 00, 91 present Proceeds to **Second Generate AC** (GAC 2) with ARPC

8A != 00 May still proceed to GAC 2 (to generate AAC)

Issuer Scripts Sends scripts to card (via APDU 80 86...)

Summary of Online Authorization

Step Description

Trigger When card generates ARQC during GAC 1

Format ISO 8583 (fields + TLV tags in Field 55)

Host Checks Card auth, ARQC validation, limits, blacklists

Host Sends ARPC, Response Code, possibly Issuer Scripts

Step	Description
Outcome	Determines how terminal proceeds in GAC 2

Perfect! Let's go **in-depth** into **Step 8: Generate AC – Second (GAC 2)** — the final cryptographic step in the EMV flow. This determines the **final outcome** of the transaction based on what the host has responded.

8. **Generate AC – Second (GAC 2) – In-Depth**

Purpose of GAC 2

After the **host responds** to the terminal's **online authorization request**, the terminal must:

1. **Pass issuer response data back to the card.**
2. Ask the card to **verify the ARPC** (Authorization Response Cryptogram).
3. Allow the card to **finalize the transaction**:
 - Approve it (TC)
 - Decline it (AAC)

This is done by sending a **second GENERATE AC (GAC 2)** command.

When GAC 2 Happens

You call **GAC 2** after:

-  Receiving a response from the issuer (host)
 -  Getting ARPC or scripts (tag 91, 71, 72)
 -  Optionally executing issuer scripts
-

GAC 2 – APDU Command Format

less

CopyEdit

CLA | INS | P1 | P2 | Lc | CDOL2 Data | Le

80 | AE | 00 | 00 | XX | [CDOL2] | 00

- CLA: 80 – proprietary
- INS: AE – GENERATE AC

- P1: 00 – Cryptogram type: **TC/AAC decision is left to the card**
 - P2: 00 – always
 - CDOL2 Data: Data defined by the card to be sent in GAC 2
 - Le: 00 – max response length expected
-

What is CDOL2?

CDOL2 = Card Data Object List 2

- A list of tags the card expects in the GAC 2 command.
- Defined during the EMV personalization.
- Might include:
 - 8A – Authorization Response Code (e.g., 00 for approved)
 - 91 – Issuer Authentication Data (ARPC)
 - 71, 72 – Issuer scripts (optional)

Some cards use an empty CDOL2 (i.e., no data expected), especially if no 91 or scripts are returned.

Example GAC 2 Command

CSS

CopyEdit

80 AE 00 00 0A

 8A 02 30 30 ; Auth Response Code: '00'

 91 08 AA BB CC ... ; ARPC (Issuer Auth Data)

00

Total 0A bytes of data (2 + 8)

GAC 2 Response from Card

Like GAC 1, the card returns:

- Final cryptogram (TC or AAC)

- Updated counters
 - Additional data
-

 **Response Format: (Either Tag 80 or 77)**

Format 1 – Template 80:

mathematica

CopyEdit

80 XX

9F27 01 40 ; CID – Cryptogram Info = TC

9F36 02 00 0F ; ATC

9F26 08 12 34 ... ; Application Cryptogram

9F10 0E ... ; Issuer Application Data

90 00

Format 2 – Template 77:

r

CopyEdit

77 XX

9F27 01 40

9F36 02 00 0F

9F26 08 12 34 ...

9F10 0E ...

90 00

 **Key Tags in GAC 2 Response**

Tag Meaning

9F27 CID – Cryptogram Info Data

9F26 Application Cryptogram (final one)

Tag Meaning

9F36 ATC – Application Transaction Counter

9F10 Issuer Application Data

9F37 Unpredictable Number

How to Interpret 9F27 – Cryptogram Info Data

CID Value	Meaning	Action
40	TC (Transaction Certificate)	Final approval
00	AAC (Application Auth Cryptogram)	Final decline
80	ARQC (should not appear here – error)	

Terminal's Final Action

- | If CID = TC (40) |  Transaction Approved → Print Receipt |
 - | If CID = AAC (00) |  Transaction Declined → Show failure |
 - | If Script Failed |  Decline or reverse (depending on rules) |
-

Common Errors

SW1 SW2 Meaning

6985	Conditions not satisfied (e.g., missing tag like 8A or 91)
6A80	Incorrect data (wrong format for CDOL2)
6700	Wrong length

Summary

Step	Description
GAC 2 Purpose	Finalize txn with issuer's response
APDU Command	80 AE 00 00 [CDOL2]

Step	Description
Output Cryptogram TC or AAC (Tag 9F27)	
CDOL2 Data	May include 8A, 91, scripts
Final Decision	Based on 9F27

Great! Now let's go **in-depth** into **Step 9: Issuer Scripts** — an optional but powerful feature in EMV that allows the **issuer (bank)** to **remotely update or control** the card's behavior during an online transaction.

9. **Issuer Scripts (if any) – In-Depth**

What Are Issuer Scripts?

Issuer Scripts are **APDU commands** sent by the **issuer bank's host** to the **EMV card via the terminal**, typically after an **online authorization (ARQC)** is received and validated.

They allow the **issuer to perform actions on the card**, such as:

- Blocking the card (e.g., if lost/stolen)
- Updating internal counters or data
- Unlocking a blocked PIN
- Deactivating an app

These scripts are returned in the **host response** (ISO 8583), wrapped in TLV tags, and **executed before or after the second Generate AC (GAC 2)**.

Tags Used for Issuer Scripts

Tag Description

71 Issuer Script Template 1 (executed **before GAC 2**)

72 Issuer Script Template 2 (executed **after GAC 2**)

86 Command to be executed (wrapped inside 71/72)

9F18 Issuer Script Identifier (optional)

 Scripts are always nested as TLVs inside tag 71 or 72.

Example of Issuer Script in Host Response

markdown

CopyEdit

72 0B

86 09

80 E2 00 00 03 12 34 56 00

Breakdown:

- 72: Script Template 2 (post-GAC2)
 - 86: Script Command
 - 80 E2: INS = PUT DATA
 - 00 00: P1 P2
 - 03: Lc = 3 bytes of data
 - 12 34 56: Data
 - 00: Le
-

 **When and How Scripts Are Executed**

◆ **Issuer → Terminal:**

- Issuer includes 71/72 TLV in the **online response**.

◆ **Terminal → Card:**

- Terminal wraps the command in an **APDU** and sends to the card.

 **Execution Command Format:**

css

CopyEdit

80 86 00 00 Lc [Script Data]

- 80: CLA (proprietary)
 - 86: INS (Execute script)
 - Lc: Length of script data (usually 9–10 bytes)
 - Script Data: Raw command (INS, P1, P2, Lc, Data, Le)
-

Example Terminal APDU to Execute Script:

CopyEdit

80 86 00 00 09

80 E2 00 00 03 12 34 56 00

Types of Issuer Scripts

Type	Tag Timing	Purpose
------	------------	---------

Script 1	71	Before GAC 2 Pre-authorized updates or status change
----------	----	---

Script 2	72	After GAC 2 Finalization actions (e.g., block card)
----------	----	--

Security of Scripts

- Scripts are **digitally authenticated** using **issuer keys**.
 - The card verifies script authenticity before executing.
 - If verification fails, the card may reject the script and **fail the transaction**.
-

Tag Summary for Scripts

Tag	Meaning
-----	---------

71	Issuer Script Template 1 (pre-GAC 2)
----	--------------------------------------

72	Issuer Script Template 2 (post-GAC 2)
----	---------------------------------------

86	Command to execute (INS like PUT DATA)
----	--

80 86 Execute Script APDU

Script Failure Consequences

- The card might:
 - Set internal failure flags.
 - Reject the second GAC (returning **AAC** instead of **TC**).

- Require a **reversal** or report script failure to the issuer.
 - Terminal logs the failure, often in tag 89 (Script Results).
-

Real-Life Uses of Issuer Scripts

Use Case	Action via Script
Hotlist stolen card	Block card via script
Update loyalty counters	PUT DATA command
PIN unblock	Update PIN try counter
App version update	App-specific script

Summary

Item	Detail
Tags	71, 72, 86, 9F18
Used in response	From issuer after online auth
Executed by	Terminal → Card via 80 86 APDU
Purpose	Card-side control/update
Security	Verified using issuer keys

Certainly! Here's a **comprehensive list of the key EMV APDU commands**, each with:

- Command format
 - Clear explanation
 - Real example
 - Common tags involved
-

EMV APDU Command Cheat Sheet with Explanations & Examples

1. SELECT AID – 00 A4 04 00

Purpose:

Used by the terminal to select a file or application on the card — most commonly, the **AID** (Application Identifier).

Command Format:

less

CopyEdit

CLA | INS | P1 | P2 | Lc | Data (AID) | Le

00 | A4 | 04 | 00 | XX | [AID] | 00

Example:

Select MasterCard AID:

CopyEdit

00 A4 04 00 07 A0 00 00 00 04 10 10 00

Response (TLV):

r

CopyEdit

6F 1E

84 07 A0 00 00 00 04 10 10

A5 ...

90 00

2. GET PROCESSING OPTIONS (GPO) – 80 A8 00 00

 **Purpose:**

Request the card's processing capabilities — returns AIP (Application Interchange Profile) and AFL (Application File Locator).

 **Command Format:**

less

CopyEdit

CLA | INS | P1 | P2 | Lc | Data (Tag 83 + PDOL Data) | Le

80 | A8 | 00 | 00 | XX | 83 [Length] [PDOL data] | 00

 **Example:**

With dummy PDOL values:

mathematica

CopyEdit

80 A8 00 00 0C 83 0A 00 00 00 00 00 00 00 00 00 00 00

 **Response:**

CopyEdit

80 06

82 00 38 → AIP

94 08 01 02 02 03 04 01 → AFL

90 00

3. READ RECORD – 00 B2 XX YY 00

 **Purpose:**

Read application data from a specific record within a specified SFI (Short File Identifier).

 **Command Format:**

bash

CopyEdit

CLA | INS | P1 (record number) | P2 (SFI << 3 | 4) | Le

00 | B2 | 01 | 0C (SFI=2) | 00

 **Example:**

Read Record 1 from SFI 2:

mathematica

CopyEdit

00 B2 01 0C 00

 **Response:**

CopyEdit

70 1A

5A 08 4761XXXXXXXXXXXXXX

5F24 03 241231

57 13 4761XXXXXXXXXXXXD2412...

90 00

4. GENERATE AC (GAC 1) – 80 AE 80 00

 **Purpose:**

Ask the card to generate a cryptogram based on CDOL1 (e.g., ARQC, AAC).

 **Command Format:**

less

CopyEdit

CLA | INS | P1 | P2 | Lc | CDOL1 Data | Le

80 | AE | 80 | 00 | XX | [Data] | 00

- P1 = 80: Request ARQC
- P1 = 00: Request TC/AAC

 **Example:**

css

CopyEdit

80 AE 80 00 1A [CDOL1 data] 00

 **Response:**

objectivec

CopyEdit

77 28

9F27 01 80 → CID (ARQC)

9F36 02 00 0C → ATC

9F26 08 AA BB ... → Cryptogram

90 00

5. GENERATE AC (GAC 2) – 80 AE 00 00

 **Purpose:**

Card finalizes the transaction based on issuer's response (TC or AAC). Sends CDOL2 data.

 **Command Format:**

css

CopyEdit

80 AE 00 00 [Lc] [CDOL2 Data] 00

 **Example:**

css

CopyEdit

80 AE 00 00 0A [8A (Auth Code) + 91 (ARPC)] 00

 **Response:**

scss

CopyEdit

77 18

9F27 01 40 → TC (approved)

9F26 08 ...

9F10 0E ...

90 00

6. GET DATA – 80 CA XX XX 00



Purpose:

Retrieve public data from the card (e.g., PIN Try Counter, ATC, last online ATC).



Command Format:

CSS

CopyEdit

80 CA [Tag1] [Tag2] 00



Examples:

- Get ATC: 9F36 →

r

CopyEdit

80 CA 9F 36 00

- Get PIN Try Counter: C0 →

r

CopyEdit

80 CA 9F 17 00



Response:

CopyEdit

9F36 02 00 15 → ATC = 21

90 00

7. EXTERNAL AUTHENTICATE – 00 82 00 00



Purpose:

Used by the terminal in **Static or Dynamic Data Authentication** to send the response cryptogram (rare in contactless flow).

8. INTERNAL AUTHENTICATE – 00 88 00 00



Purpose:

Used in **offline dynamic data authentication (DDA)**. Terminal sends a challenge, and card replies with a signed response.

9. SCRIPT EXECUTION (Issuer Script) – 80 86 00 00



Purpose:

Terminal sends **issuer scripts (Tag 71/72)** to the card.



Command Format:

CSS

CopyEdit

80 86 00 00 [Lc] [Script Command Data]



Example:

CopyEdit

80 86 00 00 09 80 E2 00 00 03 12 34 56 00

- This wraps a PUT DATA command from the issuer.



Response:

CopyEdit

90 00 → Script executed successfully



Summary Table

APDU Command Purpose	Response Tags
00 A4 04 00	SELECT AID
6F, 84, 50, 9F38	80 A8 00 00
GPO	82 (AIP), 94 (AFL)

APDU Command Purpose		Response Tags
00 B2 XX YY 00	READ RECORD	5A, 57, 5F24, 8E, 9F10, etc.
80 AE 80 00	GAC 1 (Request ARQC)	9F27, 9F26, 9F36, 9F10
80 AE 00 00	GAC 2 (Finalize – TC/AAC)	9F27, 9F26, 9F36, 9F10
80 CA XX XX 00	GET DATA	Various (e.g., 9F36, 9F17)
80 86 00 00	Execute Issuer Script	90 00 or error

Excellent question! Let's map **each key APDU command** to the **exact step** in the EMV transaction flow where it gets executed.

This will give you a **clear timeline of command execution** from card insertion to transaction completion.

EMV Flow – APDU Commands by Step

Step No.	Step Name	APDU Command	Purpose / Notes
1	Card Insertion & ATR	— (ATR from card)	Card responds with ATR automatically (not APDU-initiated)
2	Application Selection	00 A4 04 00 (SELECT AID)	Terminal selects app by AID
3	Get Processing Options (GPO)	80 A8 00 00	Requests AIP and AFL to determine card capabilities
4	Read Application Data	00 B2 XX YY 00 (READ RECORD)	Uses AFL to read relevant records
5	Cardholder Verification (CVM)	—	No APDU; uses CVM list (8E tag) read in previous step
6	Terminal Risk Management	80 CA 9F17 00, etc. (optional)	May use GET DATA for ATC, PIN Try Counter
7	Generate AC – First (GAC 1)	80 AE 80 00	Card generates ARQC or AAC based on CDOL1
8	Online Authorization	— (host communication)	Terminal sends ARQC + EMV data to issuer over ISO 8583
9	Issuer Script Execution (Pre-GAC2)	80 86 00 00 (Tag 71)	Terminal sends Issuer Script 1 before GAC 2
10	Generate AC – Second (GAC 2)	80 AE 00 00	Final cryptogram from card (TC or AAC) using CDOL2

Step No.	Step Name	APDU Command	Purpose / Notes
11	Issuer Script Execution (Post-GAC2)	80 86 00 00 (Tag 72)	Script execution after transaction is approved/declined
12	Transaction Completion	—	Based on result from GAC 2 → show result, print receipt

 **Additional (Optional) Commands and Where They're Used:**

APDU	When Used	Purpose
80 CA 9F36 00	Terminal Risk Management / Logging	Read Application Transaction Counter
80 CA 9F13 00	Logging / Risk Management	Last Online ATC
80 CA 9F17 00	Risk Management	Get PIN Try Counter
00 88 00 00	During Dynamic Data Authentication (DDA)	INTERNAL AUTHENTICATE (with challenge)
00 82 00 00	During Static Data Authentication (SDA)	EXTERNAL AUTHENTICATE (rare)

 **Summary Timeline**

SCSS

CopyEdit

[Insert Card]

↓

[ATR]

↓

[00 A4 04 00] – SELECT AID

↓

[80 A8 00 00] – GPO (get AFL, AIP)

↓

[00 B2 XX YY 00] – READ RECORD (get CVM list, card data)

↓

[CVM Decision] (uses CVM List from tag 8E)

↓

[Terminal Risk Checks + GET DATA]

↓

[80 AE 80 00] – GAC 1 (get ARQC or AAC)

↓

[Online Auth – Host communication]

↓

[Issuer Script Execution – tag 71 → 80 86 00 00]

↓

[80 AE 00 00] – GAC 2 (get TC or AAC)

↓

[Issuer Script Execution – tag 72 → 80 86 00 00]

↓

[Finish Transaction]

Let's go **in-depth** into the **most critical EMV tags** you've mentioned: 95, 9F33, CTQ, TTQ, 82, and 9F5D. These are key to understanding how **capabilities and risk decisions** are exchanged between the terminal and card.

Tag 95 – Terminal Verification Results (TVR)

Purpose:

95 holds **5 bytes** that track conditions the **terminal checked or encountered** during the transaction.

It helps decide **whether to go online, decline, or allow offline approval**.

Structure:

TVR = 5 bytes = 40 bits

Each bit represents a specific check or condition:

Byte	Bit	Meaning
1	b8	Offline data authentication not performed
	b7	SDA failed
	b6	ICC data missing
	b5	Card appears on terminal exception file
	b4	DDA failed
	b3	CDA failed
2	b8	ICC and terminal have different application versions
	b7	Expired application
	b6	Application not yet effective
	b5	Requested service not allowed
	b4	New card

3–5 Various CVM failure, amount over limit, merchant forced online, etc.

Example:

CopyEdit

95 05 80 00 00 00 00

- Byte 1 = 80 → Offline data auth not performed (b8)
-

Tag 9F33 – Terminal Capabilities

Purpose:

9F33 tells the card **what the terminal can do**:

Supports **magstripe, offline/online PIN, CDA, contactless**, etc.

Structure:

3 bytes — each byte has different flags.

Byte Description

- 1 Card data input: keypad, magstripe
 - 2 CVM capabilities: offline/online PIN
 - 3 Security and transaction type support
-

Example:

CopyEdit

9F33 03 E0 F0 C8

Breakdown:

- Byte 1 = E0: Terminal supports magstripe, contact chip
- Byte 2 = F0: Online & offline PIN supported
- Byte 3 = C8: Supports SDA/DDA/CDA, supports cash

This is **used in CVM decision logic and risk management**.

CTQ – Card Transaction Qualifiers (9F66)

Purpose:

Tells terminal **what features the card supports** in contactless flow (e.g., PayWave, PayPass).

Often used in **contactless-only** paths (EMV Contactless Entry Point v2.6+).

CTQ Structure (2 bytes):

Bit Feature

b8 EMV contactless supported

b7 magstripe contactless supported

b6 Online cryptogram supported

b5 Offline data authentication supported

b4 Issuer update processing supported

b3 Consumer device CVM supported

b2 Online PIN supported

Example:

CopyEdit

9F66 04 36 00 00 00

- Card supports:
 - EMV contactless
 - magstripe mode
 - online cryptogram
-

TTQ – Terminal Transaction Qualifiers (9F66 from terminal)

Purpose:

Tells card **what the terminal supports** during a contactless transaction.
It's the **mirror of CTQ**, but sent from the terminal to the card.

Used during **GPO or tap interaction** in contactless flow.

Structure (4 bytes):

Byte Purpose

- 1 Contactless modes: EMV / magstripe
 - 2 CVM support (signature, online PIN, no CVM)
 - 3 EMV risk management (floor limit, velocity check, etc.)
 - 4 Additional data flags (e.g., fast decline)
-

Example:

CopyEdit

9F66 04 36 00 C0 00

- Byte 1 = 36: Supports EMV + magstripe
 - Byte 2 = 00: No CVM
 - Byte 3 = C0: Supports floor limit + velocity check
 - Byte 4 = 00: No additional features
-

Tag 82 – AIP (Application Interchange Profile)

Purpose:

Tells terminal **what the card supports** after **GPO** (Get Processing Options).

AIP Structure (2 bytes):

Byte Bit Meaning

- 1 b8 Offline Static Data Authentication (SDA) supported
 b7 Offline Dynamic Data Authentication (DDA) supported
 b6 Cardholder verification supported
 b5 Terminal risk management supported
 b4 Issuer authentication supported
- 2 b8 Combined DDA/AC (CDA) supported
-

Example:

CopyEdit

82 02 38 00

- Byte 1 = 38 = SDA, DDA, CVM supported
 - Byte 2 = 00 = CDA not supported
-

ASI – Application Selection Indicator (Part of AID / not a tag)

Purpose:

This is a **1-byte value** appended to the AID in the terminal's AID list, used during **application selection**.

Defined in the terminal config, not sent as an APDU tag.

ASI Value Meaning

- 00 Try exact match only
01 Try partial match
02 Try exact match, else partial

Terminal uses it to determine **how strictly to match AIDs** from card vs. terminal config.

 **How They All Fit Together in EMV Flow:**

Tag	Step Used	Direction	Purpose
95	Before GAC 1	Terminal → Card	Tells what checks failed/passed
9F33	GPO & CVM	Terminal → Card	Terminal capability descriptor
9F66 (CTQ & TTQ)	Contactless Bidirectional		Declares contactless feature support
82	GPO Response	Card → Terminal	Card capabilities for offline/online
ASI	App Selection	Terminal config	Match mode for AID comparison

