

Great question. Let's dive deep into **Issuer Scripts** – a critical part of EMV (chip card) transactions that's often misunderstood.

◆ What Are Issuer Scripts?

Issuer Scripts are commands sent by the **issuer (bank)** to the **chip card** during a transaction to perform post-authorization actions like:

- Blocking the card
- Updating card data (e.g., PIN try counter)
- Changing keys or parameters
- Disabling applications

They're defined in the EMV specification (Book 3) and used mainly in **online-authorized EMV transactions**.

◆ When Are Issuer Scripts Used?

During a typical EMV transaction:

1. Card is inserted → Terminal performs **offline data authentication**, reads data.
 2. Terminal sends an **authorization request** to the issuer (MTI 0200).
 3. **Issuer may include Issuer Script Commands** in its **authorization response (MTI 0210)**.
 4. Terminal **sends the script to the card**.
 5. Card executes the script.
 6. Terminal records the outcome in **Field 55** in the **completion advice (0220/0230)** message (if needed).
-

◆ Where Do Scripts Come From?

Issuer scripts are created by the **issuer host system** and included in the **authorization response message** as part of **Tag 71** and **Tag 72** inside **Field 55**.

◆ Types of Issuer Scripts

Tag	Script Type	Description
71	Issuer Script Template 1	Sent in 1st GENERATE AC (ARQC) response
72	Issuer Script Template 2	Sent in 2nd GENERATE AC (TC/ARPC) response

Each of these can contain one or more **script commands** (with Tag + Length + Value TLV format).

◆ Examples of Issuer Script Commands

Inside Tag 71/72, you'll see TLV encoded commands like:

Tag	Description
86	Proprietary script command
87	Proprietary script command
9F18	Issuer Script Identifier
9F17	PIN Try Counter
9F10	Issuer Application Data (often used to trigger behaviors)

▣ Sample Script (Hex) in Tag 71 or 72:

cpp

CopyEdit

```
71 12          // Tag 71, Length 18 bytes
86 10 A0000000031010FFFFFFFFFFFFFFFF // Script Command (e.g., block app)
```

◆ When Exactly Scripts Are Updated?

Scripts are updated **during an EMV online transaction**, specifically:

- **During Authorization Response (MTI 0210)**, the host (issuer) adds issuer scripts inside Field 55.
- The **terminal parses and sends them to the card** using **EXTERNAL AUTHENTICATE** or other APDU commands.
- Card **executes the script commands** (e.g., to block card, update counters, etc.).

- Outcome is reported in Field 55 of the **Advice or Completion message** (e.g., MTI 0220/0230).

◆ Real Life Use Cases

Purpose	Example
Card Blocking	Issuer sees fraud risk, sends script to disable card immediately.
Counter Update	Issuer resets PIN try counter or updates offline counters.
Parameter Update	Update keys or parameters on chip for security rotation.
Emergency PIN change	Issuer forces a new PIN load during transaction.

◆ Summary: Flow

yaml

CopyEdit

Terminal --> Host (MTI 0200)

|

v

Host returns MTI 0210 with issuer scripts (Tag 71/72 in Field 55)

|

v

Terminal sends scripts to card (APDU: EXTERNAL AUTHENTICATE, etc.)

|

v

Card executes script and returns status

|

v

Terminal logs result → sends MTI 0220/0230 if needed

Would you like a **step-by-step trace** of an EMV transaction showing these exact fields, APDUs, and sample hex data?

Or a **decoded example** of Tag 71/72 in a real response message?

You said:

yes

ChatGPT said:

Let's dive deep into **Issuer Scripts** in EMV transactions:





◆ What are Issuer Scripts?

Issuer Scripts are special commands sent by the **issuer (card-issuing bank)** during an online EMV transaction. These scripts allow the issuer to **remotely update certain data on the card** after the transaction completes successfully.

They are part of **post-issuance personalization** — i.e., updating card parameters after it's already in the field.

◆ Purpose of Issuer Scripts

Issuer scripts can be used to:

Purpose	Description
 Block a card	Update card to be blocked/disabled.
 Change PIN rules	Change PIN try counter or offline PIN.
 Update CVR/IAC	Modify risk parameters or CVR (Card Verification Results).
 Other personalization	Change application data like offline limits, etc.

◆ When do Issuer Scripts Occur?

1. During **online authorization**, if the issuer wants to change something on the card, it includes script data in its **Authorization Response (ISO 8583)** message.
2. During **second Generate AC (GAC 2)** command, the terminal reads the script and executes it.

3. The terminal sends the script commands to the card **after** the online authorization and **before** completing the transaction.

◆ Tags Used for Issuer Scripts

Tag Description

71 Issuer Script Template 1 (sent **before** final GENERATE AC)

72 Issuer Script Template 2 (sent **after** final GENERATE AC)

Both templates contain one or more **script commands**, each consisting of:

- **CLA (Class byte)** – Always 0x80 or 0x84 (proprietary),
- **INS (Instruction byte)** – E.g., 0xE2 for Update Binary,
- **P1/P2** – Address or reference,
- **Lc** – Length of data,
- **Data** – Actual data to update.

◆ Example of Issuer Script (Tag 71)

hex

CopyEdit

```
71 0A 80 E2 00 00 03 12 34 56
```

This means:

- 71 0A: Tag 71, 10 bytes of script
- 80: CLA – proprietary
- E2: INS – Update Binary
- 00 00: Offset
- 03: Lc (length = 3 bytes)
- 12 34 56: Data to write to the card

◆ Where Issuer Scripts Come From

- The **issuer host system** generates them during **online processing**, based on:

- Card status
- Fraud rules
- Customer behavior
- Card expiry or personalization requirements

They are inserted into the **ISO8583 authorization response**, typically using DE 55 (EMV Data) or private fields.

◆ Execution Flow of Issuer Scripts

text

CopyEdit

1. Terminal sends ONLINE request → Host receives request
2. Host approves and includes Issuer Script (Tag 71/72) in response
3. Terminal receives response
4. Terminal sends Issuer Script to card (via APDUs)
5. Card executes script
6. Terminal completes transaction

◆ Summary Table

Step	Role	Action
1	Terminal	Sends online request
2	Issuer	Decides if script is needed
3	Issuer	Adds Tag 71/72 in ISO 8583 response
4	Terminal	Parses response, sends script to card
5	Card	Executes commands
6	Terminal	Completes transaction

Would you like a **real-world example with full ISO8583 and APDU breakdown including issuer script handling?**

You said:

yes

ChatGPT said:

Perfect — let's walk through a **real-world EMV online transaction** that includes **Issuer Scripts**, showing how they're constructed, transferred, and executed.

 **Scenario: Issuer wants to update a CVR bit on the card (e.g., block offline transactions).**

Step-by-Step Transaction Flow (with APDUs, ISO 8583, and Scripts)

◆ **Step 1: Terminal sends first GENERATE AC (GAC 1)**

Used to request an **ARQC (Authorization Request Cryptogram)**.

plaintext

CopyEdit

Command: 80 AE 80 00 1C

Data:

- CDOL1 contents (e.g., amount, date, unpredictable number, etc.)

Response: 77 xx ... 9F26 (ARQC) ...

◆ **Step 2: Terminal sends Online Authorization Request (ISO 8583)**

The ARQC and other EMV data are packaged in **DE55** or **private fields**:

plaintext

CopyEdit

Field 55 (EMV TLV data):

9F26 08 <ARQC>

9F02 06 <Amount>

9F1A 02 <Country Code>

...

This request is sent to the **issuer host**.

◆ Step 3: Issuer Analyzes and Responds (includes Issuer Script)

Issuer decides to **send a script** to update the card (e.g., block offline use by updating CVR).

Sample Issuer Script in Response:

hex

CopyEdit

Field 55 response includes:

71 0A 80 E2 00 00 03 12 34 56

Which means:

- 71: Issuer Script Template 1 (to run **before** 2nd GAC)
- 0A: Length
- 80 E2 00 00 03 12 34 56 → Update Binary at offset 0000, write 3 bytes 12 34 56

Also, the issuer may return:

plaintext

CopyEdit

8A 02 30 30 // Authorization Response Code = "00" (Approved)

◆ Step 4: Terminal Parses Response and Sends Script to Card

It sends an **APDU** to execute the script:

plaintext

CopyEdit

Command: 80 E2 00 00 03

Data: 12 34 56

Response: 9000 (Success)

This shows that the card **accepted the issuer script**.

◆ Step 5: Terminal sends second GENERATE AC (GAC 2)

This completes the transaction and includes issuer authentication data:

plaintext

CopyEdit

Command: 80 AE 00 00 1C

Data: CDOL2 values

Response: 77 xx ... 9F27 (CID) ... 9F36 (ATC) ...

◆ Step 6: Transaction Completed

- If **GAC2 returns AAC**, transaction fails.
- If **GAC2 returns TC (Transaction Certificate)**, it's successful.

Terminal prints receipt, closes connection.

✅ Final Summary

Component	Value
Script Template	71 (Before GAC2)
Script Command	80 E2 00 00 03 12 34 56
ISO8583 Response	Contains Field 55 with 8A, 71, etc.
Terminal Action	Sends 80 E2 ... to card
Card Action	Executes it, returns 9000
Completion	Second GAC completes transaction