How to read a Credit card via NFC

Smart card application protocol data unit

In the context of smart cards, an **application protocol data unit** (**APDU**) is the communication unit between a smart card reader and a smart card. The structure of the APDU is defined by ISO/IEC 7816-4 *Organization, security, and commands for interchange*

APDU message command-response pair

There are two categories of APDUs: command APDUs and response APDUs.

A command APDU is sent by the reader to the card – it contains a mandatory 4-byte header (CLA, INS, P1, P2) and from 0 to 65 535 bytes of data.

A response APDU is sent by the card to the reader – it contains from 0 to 65 536 bytes of data, and 2 mandatory status bytes (SW1, SW2). The card will execute the command and send a response APDU back to the terminal. The response APDU has an optional body consisting of data and a mandatory trailer with two status bytes “SW1” and “SW2”. SW1 and SW2 combined are the status word (SW). If the status word has the value 0x9000 (SW1 = 0x90, SW2=0x00), the command was successfully executed by the card.

Command APDU		
Field name	**Length (bytes)**	**Description**
CLA	1	Instruction class - indicates the type of command, e.g. interindustry or proprietary
INS	1	Instruction code - indicates the specific command, e.g. "write data"
P1-P2	2	Instruction parameters for the command, e.g. offset into file at which to write the data
L_c	0, 1 or 3	Encodes the number (N_c) of bytes of command data to follow 0 bytes denotes N_c=0 1 byte with a value from 1 to 255 denotes N_c with the same length 3 bytes, the first of which must be 0, denotes N_c in the range 1 to 65 535 (all three bytes may not be zero)
Command data	N_c	N_c bytes of data
L_e	0, 1, 2 or 3	Encodes the maximum number (N_e) of response bytes expected 0 bytes denotes N_e=0 1 byte in the range 1 to 255 denotes that value of N_e, or 0 denotes N_e=256 2 bytes (if extended L_c was present in the command) in the range 1 to 65 535 denotes N_e o f that value, or two zero bytes denotes 65 536 3 bytes (if L_c was not present in the command), the first of which must be 0, denote N_e in the same way as two-byte L_e
Response APDU		
Response data	N_r (at most N_e)	Response data
SW1-SW2 (Response trailer)	2	Command processing status, e.g. 90 00 (hexadecimal) indicates success

to parse the response for visualizing, you can use this tool

Table of the most important Smart Card Commands

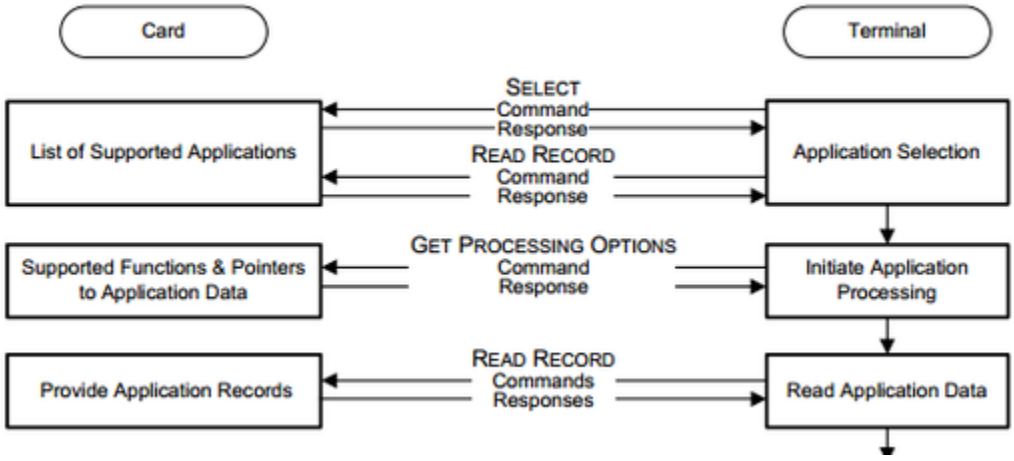
The full table could be found here

Table of the most important Smart Card Return codes

The full table could be found here and here

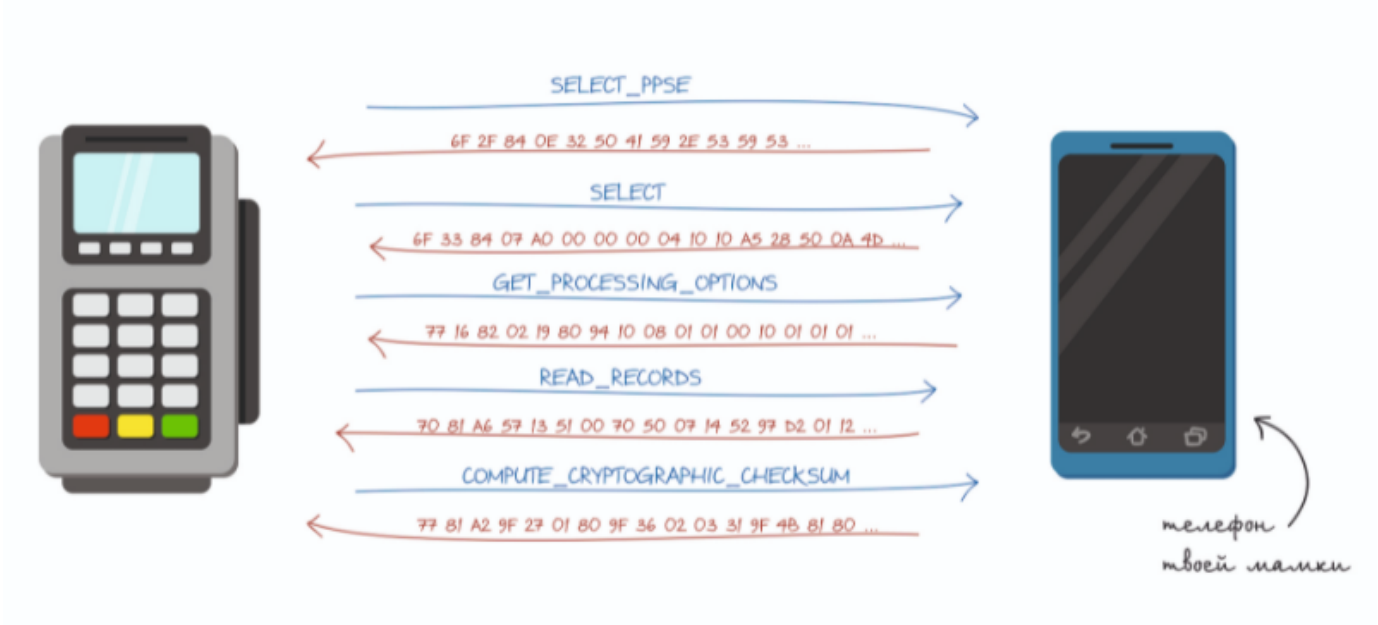
Read the Credit card via NFC

Simple Read Card Flow



The transaction processing process for contactless cards is trimmed compared to chip cards and usually works in the following mode:

1. The terminal sends the command **SELECT PPSE** (Proximity Payment System Environment). The card sends a list of supported applications.
2. The terminal sends a **SELECT** command. In response, receives the necessary details of the application.
3. The terminal sends the **GET_PROCESSING_OPTIONS** command. The card answers what type of authentication it supports and whether there is verification of the cardholder there.
4. The terminal sends the **READ_RECORDS** command. The card in response sends Track1 and Track2 almost the same as that recorded on the magnetic stripe of the card.
5. The terminal sends the **COMPUTE_CRYPTOGRAPHIC_CHECKSUM** command. This means that the card should, based on the transmitted Unpredictable Number, generate the value of CVC3.



Example detail

- Step 1: **SELECT PPSE** (starting the communication).

With the contactless card structure, we have to select the Proximity Payment System Environment (PPSE) if there is any, to start the process. To do this, we have to create a command using the APDU header and trailer which are specified in the ISO-7816-4.

the command should be ***apdu*** = ***SELECT*** + *[len(**DF_PPSE**)]* + ***DF_PPSE*** + *[0x00]*

	SELECT						
Header	**CLA**	**INS**	**P1**	**P2**	**Lc**	**PPSE**	**Le**
value	00	A4	04	00	0E	32 50 41 59 2E 53 59 53 2E 44 44 46 30 31 00	00
explain	class Byte	select command	Select by name	The first record	lenght(PPSE) = 14	value of 2PAY.SYS.DDF01	

```
Select PPSE Application
send: 00 A4 04 00 0E 32 50 41 59 2E 53 59 53 2E 44 44 46 30 31 00
resp: 6F 39 84 0E 32 50 41 59 2E 53 59 53 2E 44 44 46 30 31 A5 27 BF 0C
24 61 22 4F 07 A0 00 00 00 03 10 10 50 10 56 43 42 20 56 49 53 41 20 50
41 59 57 41 56 45 87 01 01 9F 2A 01 03 90 00
```

response detail explanation

```
6F 39 -- File Control Information (FCI) Template
         84 0E -- Dedicated File (DF) Name
               32 50 41 59 2E 53 59 53 2E 44 44 46 30 31 (BINARY)
         A5 27 -- File Control Information (FCI) Proprietary Template
               BF 0C 24 -- File Control Information (FCI) Issuer
Discretionary Data
                        61 22 -- Application Template
                              4F 07 -- Application Identifier (AID) -
card
                                 A0 00 00 00 03 10 10 (BINARY)
                              50 10 -- Application Label
                                 56 43 42 20 56 49 53 41 20 50 41
59 57 41 56 45 (=VCB VISA PAYWAVE)
                              87 01 -- Application Priority Indicator
                                 01 (BINARY)
                              9F 2A 01 -- The value to be appended to
the ADF Name in the data field of the SELECT command, if the Extended
Selection Support flag is present and set to 1
                                    03 (BINARY)
    90 00 -- Command successfully executed (OK)
```

from the response to this request, now, we have an Application Identifier(AID = *A0000000031010*): this identification means what type of card and what company(Visa, Mastercard, Amex, etc.) is assigned to. Also from which country and type of technology. AIDs are very important to keep digging to get more information.

- Step 2: **SELECT AID** command to obtain information with that AID

```
apdu = SELECT + [len(AID)] + AID+ [0x00]
```

	SELECT AID						
Header	**CLA**	**INS**	**P1**	**P2**	**Lc**	**AID**	**Le**
value	00	A4	04	00	07	A0000000031010	00
explain	class Byte	select command	Select by name	The first record	lenght(A0000000031010) = 7	the selected AID	

```
send: 00 A4 04 00 07 A0 00 00 00 03 10 10 00
resp: 6F 4B 84 07 A0 00 00 00 03 10 10 A5 40 50 10 56 43 42 20 56 49 53
41 20 50 41 59 57 41 56 45 87 01 01 5F 2D 02 65 6E 9F 38 18 9F 55 01 9F
05 09 9F 03 06 9F 1A 02 95 05 5F 2A 02 9A 03 9C 01 9A 35 06 BF 0C 08 9F
5A 05 40 07 04 07 04 90 00
```

response explanation

```
6F 4B -- File Control Information (FCI) Template
         84 07 -- Dedicated File (DF) Name
               A0 00 00 00 03 10 10 (BINARY)
         A5 40 -- File Control Information (FCI) Proprietary Template
               50 10 -- Application Label
                  56 43 42 20 56 49 53 41 20 50 41 59 57 41 56 45
(=VCB VISA PAYWAVE)
               87 01 -- Application Priority Indicator
                  01 (BINARY)
               5F 2D 02 -- Language Preference
                     65 6E (=en)
               9F 38 18 -- Processing Options Data Object List (PDOL)
                     9F 55 01 -- Terminal Transaction Qualifiers
                     9F 05 09 -- Amount, Authorised (Numeric)
                     9F 03 06 -- Amount, Other (Numeric)
                     9F 1A 02 -- Terminal Country Code
                     95 05 -- Terminal Verification Results (TVR)
                     5F 2A 02 -- Transaction Currency Code
                     9A 03 -- Transaction Date
                     9C 01 -- Transaction Type
                     9A 35 06 -- Unpredictable Number
               BF 0C 08 -- File Control Information (FCI) Issuer
Discretionary Data
                     9F 5A 05 -- [UNKNOWN TAG]
                           40 07 04 07 04 (BINARY)
    90 00 -- Command successfully executed (OK)
```

- Step 3: **GET_PROCESSING_OPTIONS**

	Get Processin Option						
Header	CLA	INS	P1	P2	Lc	DATA	Le
value	80	A8	00	00	23	23 83 21 28 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 50 00 00 00 00 00 00 09 78 21 10 01 00 33 8E E6 B7	00

```
send: 80 A8 00 00 23 83 21 28 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 02 50 00 00 00 00 00 09 78 21 10 01 00 33 8E E6 B7 00
resp: 77 52 82 02 00 00 94 04 08 04 04 00 57 13 45 24 04 18 60 30 00 11
D2 50 42 01 16 68 22 25 00 00 0F 5F 20 02 20 2F 5F 34 01 00 9F 10 07 06
01 0A 03 A0 00 00 9F 26 08 A6 04 49 6F 8A 20 8C 31 9F 27 01 80 9F 36 02
00 2A 9F 6C 02 28 00 9F 6E 04 20 70 00 00 90 00
```

response explanation:

```
77 52 -- Response Message Template Format 2
          82 02 -- Application Interchange Profile
                00 00 (BINARY)
          94 04 -- Application File Locator (AFL)
                08 04 04 00 (BINARY)
          57 13 -- Track 2 Equivalent Data
                45 24 04 18 60 30 00 11 D2 50 42 01 16 68 22 25
                00 00 0F (BINARY)
          5F 20 02 -- Cardholder Name
                   20 2F (= /)
          5F 34 01 -- Application Primary Account Number (PAN) Sequence
Number
                   00 (NUMERIC)
          9F 10 07 -- Issuer Application Data
                   06 01 0A 03 A0 00 00 (BINARY)
          9F 26 08 -- Application Cryptogram
                   A6 04 49 6F 8A 20 8C 31 (BINARY)
          9F 27 01 -- Cryptogram Information Data
                   80 (BINARY)
          9F 36 02 -- Application Transaction Counter (ATC)
                   00 2A (BINARY)
          9F 6C 02 -- Mag Stripe Application Version Number (Card)
                   28 00 (BINARY)
          9F 6E 04 -- Visa Low-Value Payment (VLP) Issuer Authorisation
Code
                   20 70 00 00 (BINARY)
90 00 -- Command successfully executed (OK)
```

- get Left PIN try

```
send: 80 CA 9F 17 00
resp: 9F 17 01 03 90 00
```

response explanation:

```
9F 17 01 -- Personal Identification Number (PIN) Try Counter
            03 (BINARY)
    90 00 -- Command successfully executed (OK)
```

Source code example: