Decorvet, Elias

Smartcards

Table of Contents

[Protokolle 2](#_Toc47514492)

[T0-Protokoll 2](#_Toc47514493)

[Übersicht 2](#_Toc47514494)

[Funktionen (TPDU) 2](#_Toc47514495)

[Applikations Kommuikation (APDU) 3](#_Toc47514496)

[Filesystem API 4](#_Toc47514497)

[Generell 4](#_Toc47514498)

[Hirarchie 5](#_Toc47514499)

[FS-Funktionen 5](#_Toc47514500)

[Transparente Files 5](#_Toc47514501)

[Record Files 6](#_Toc47514502)

# Protokolle

## T0-Protokoll

### Übersicht

* Byte orientiert
* Error-Detection wird durch ein Paritäts-bit sichergestellt
  + Receiver verlangt Nachsendung
  + IO-Line 🡺von High zu Low
* Daten werden durch TPDU-Pakete gesendet
  + Befehl von Reader zur Karte
  + Befehl von Karte zum Reader
* Befehlsheader hat fünf Felder in der länge von je einem Byte
  + CLA
    - Klassendesignation um ein Instructionset aufzurufen
  + INS
    - Deutet auf eine spezifische Instruktion aus dem gewählten Instructionset
  + P1
    - Spezifiziert die Addressierungsart der Instruktion (CLA 🡺 INS 🡺 P1)
  + P2
    - Spezifiziert ebenfalss die Addressierungsart der Instruktion
  + P3
    - Spezifiziert die Anzahl an Daten-Bytes zur/von der Smartcard als Teil der (CLA🡺INS) Ausführung

### Funktionen (TPDU)

|  |  |
| --- | --- |
| CLA-Byte (Hex) | Instructionset |
| 0X | ISO-7816-4 Befehle (Dateien und Security) |
| 10 - 7F | Für zukünftigen Gebrauch |
| 8X | ISO-7816-4 Befehle (Generelles) |
| AX | Applikations- / Verkäufer-Spezifische Instruktionen |
| B0 - CF | ISO-7816-4 Befehle (Generelles) |
| D0 - FE | Applikations- / Verkäufer-Spezifische Instruktionen |
| FF | Reserviert für Protokoll-Auswahl |

|  |  |  |  |
| --- | --- | --- | --- |
| INS-Wert | Befehlsname | INS-Wert | Befehlsname |
| 0E | Erase Binary | C0 | Get Response |
| 20 | Verify | C2 | Envelope |
| 70 | Manage Channel | CA | Get Data |
| 82 | External Authenticate | D0 | Write Binary |
| 84 | Get Challange | D2 | Write Record |
| 88 | Internal Authenticate | D6 | Update Binary |
| A4 | Select File | DA | Put Data |
| B0 | Read Binary | DC | Update Record |
| B2 | Read Records | E2 | Append Record |

|  |  |
| --- | --- |
| TDPU Antwort | Bedeutung |
| ACK | Karte hat den CLA 🡺 INS Befehl erhalten |
| NULL | Karte verarbeitet den vorherigen Befehl noch (Reader muss warten) |
| SW1 | Status-Antwort des jetzigen Befehls |
| SW2 | (Optional) Weitere Status-Antwort |

### Applikations Kommuikation (APDU)

#### Anfrage

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Header | | | |  | | |
| CLA | INS | P1 | P2 | Body | | |
|  | | | | Le-Field | Data field | Le fiedl |

#### Antwort

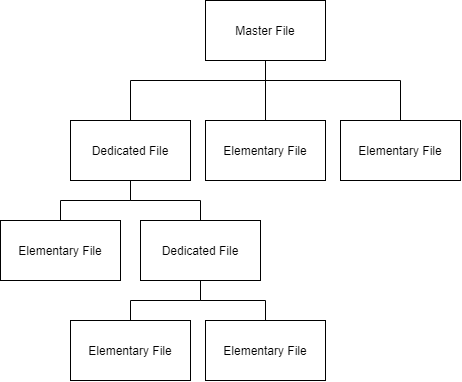
|  |  |
| --- | --- |
| Body | |
| Data Field | SW1 (&SW2) |

# Filesystem API

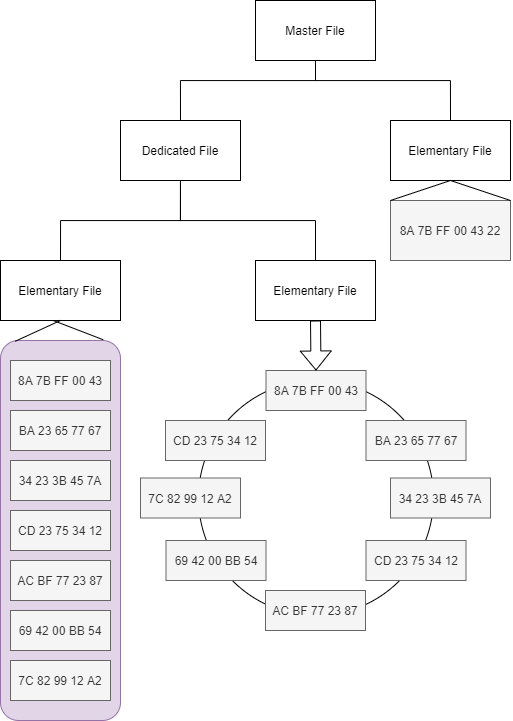
## Generell

* Filesystem wird auf einem EEPROM (Electrically Eraseable Programmable Read Only Memory) verwendet
* Hirarchische Struktur
* Alle Files müssen einen *einzigartigen* Pfad besitzen
* Drei File-Typen, die mit einem 2Byte langen Identifier dargestellt werden
  + Master File
    - Root für die Datenstruktur
    - ID: 3F00
  + Dedicated / Directory File
    - Container für weitere Container & Elementary-Files
  + Elementary File
    - Transparent File
      * Binär-String (unstrukturiertes bin-File)
        + Zum Lesen wird ein Offset und die Länge des zu lesenden Strings angegeben
    - Linear, Fixed-Length Record
      * Records, die durch eine Sequenznummer identifiziert werden
        + Alle Records haben die *gleiche* Länge in Bytes
    - Linear, Variable-Length Record
      * Records, die durch eine Sequenznummer identifiziert werden
        + Records können *verschiedene* Längen in Bytes aufweisen
        + Haben höheren Overhead als Fixed-Length Records
    - Cyclic, Fixed-Length Record
      * Kann Records transparent darstellen
      * Man kann es sich wie einen Kreis aus Files vorstellen

## Hirarchie



## Daten Struktur im Filesystem



## FS-Funktionen

### Transparente Files

#### Select File

* Erstellt einen logischen Pointer zu einem spezifischen File
* Der Pointer *wird* für File-Manipulation *benötigt*
* Smartcard Filesystem ist *nicht* Multithread, es *können* aber mehrere Pointers gleichzeitig existieren
  + Kann durch Manage Channel-Befehl kreiert werden (Mehre logische Kanäle zwischen Reader & Karte)
* Identifikation kann durch mehrere Arten geschehen
  + File Identifier (2-Byte Wert)
  + DF-Name (String aus Bytes)
  + Path (mehrere, aneinander gereihte File Identifiers)
  + Short ID
* (Nicht alle Smartcards unterstützen diese ID-Möglichkeiten)

#### Read Binary

* Wird vom Reader verwendet um Teile eines Elementary Files zu lesen
* Muss auf *transparente* Files angewendet werden
* Kann *nicht* mit Record-Files arbeiten
* Benötigt einen Offset und eine Länge der zu lesenden Bytes

#### Write Binary

* Wird verwendet um Daten als Byte-Stream auf einer Smartcard zu speichern
* Muss auf ein *transparentes* File angewendet werden

#### Update Binary

* Eine Reader-Apllikation kann direkt ein File abändern
* Muss auf ein *transparentes* File angewendet werden
* Ist wie die Write-Binary-Funktion, aber dazu ausgelegt, Daten an einem bestimmten Punkt zu *überschreiben*
* Benötigt einen Offset und die Länge des zu überschreibenden Byte-Strings

#### Erase binary

* Wird verwendet um Bytes aus dem spezifizierten File zu löschen
* Muss auf ein transparentes File angewendet werden
* Benötigt einen Offset und eine Länge des zu entfernenden Byte-Strings

### Record Files

#### Read Record

* Wird verwendet um einen oder mehrere Records aus einem Elementary File zu lesen
* Muss auf ein *record-orientiertes* File angewendet werden
* Folgende Daten können je nach Parametern ausgegeben werden:
  + Ein spezifischer Record
  + Alle Records vom Anfang des Files bis zu einem spezifischen Record
  + Alle Records ab einem spezifischen Record bis zum Ende des Files

#### Write Record

* Wird verwendet um einen Record zu schreiben
* Kann einen Record überschreiben oder einen neuen Record hinzufügen
* Muss auf ein *record-orientiertes* File angewendet werden

#### Append Record

* Wird verwendet um einen neuen Record an ein File anzufügen
* Muss auf *record-orientierte* Files angewendet werden
* Kann in einem linear-record-oriented File einen Record am Ende ansetzen
* Kann in einem cyclic-record-oriented File einen neuen Record am Anfang hinzufügen

#### Update Record

* Wird verwendet um einen spezifischen Record in einem Elementary File zu bearbeiten
* Muss auf einem *record-oriented* File angewendet werden

#### Get Data

* Liest Daten aus einem Daten-Objekt
* Datenobjekt-Struktur ändert je nach Hersteller

#### Put Data

* Speichert Daten vom Reader zu einem Daten-Objekt auf der Smartcard
* Dieser Befehl ist *Kartenspezifisch*