What to Do With APDU?

Sebastian Hans

Sr. Tech. Expert
Sun Microsystems, Inc.
http://www.sun.com

Hartti Suomela

Sr. Tech. Expert
Forum Nokia
http://forum.nokia.com

TS-5642

In the Next 60 Minutes…

We will present an overview of the APDU package of SATSA API (Java™ Specification Request (JSR) 177) that can be used to access smart cards.

Agenda

Overview of Smart Cards and SIM

Overview of SATSA API (JSR 177)

Using SATSA-APDU

  Control flow, APDUs, access control

SATSA APDU on Nokia devices

DEMO

Smart Cards

- A tamper-resistant computer with an operating system
- Secure embedded microchip and secure memory
  - 8- to 32-bit CPU, 32–256 KB EEPROM
  - Larger memory uses FLASH
- Only one I/O line per interface
  - Contact, contactless communication or a combination
- Used in:
  - Banking (payment, credit-debit)
  - SIM cards in GSM phones
  - Health insurance
  - Ticketing and Transport
  - In short, secure processing and secure storage
- Billions of cards are used worldwide

SIM (Subscriber Identity Module)

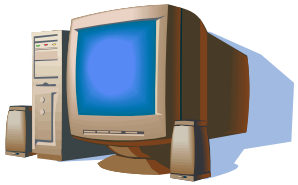
- Mandatory in every GSM/UMTS phone
- R-UIM (Removable User Identity Module) in CDMA
- Removable smart card
  - Identifies the mobile phone “user” and gives secure access to the GSM/UMTS network
  - Enables the roaming between the different networks
  - Stores:
    - Configuration settings for network connections
    - Authentications keys
    - User information
    - Possibly phone number (operator dependent)
  - Issued and managed by the operator
- Most SIM cards are Java Card™ based

What Is a Java Card API?

- A subset of the Virtual Machine for the Java platform (JVM™ machine) and programming language for smart cards
- Secure application platform that allows:
  - Run several isolated smartcard applications in parallel in the card
  - Load and manage dynamically applications in the card
  - A secure communication of applications in the card
  - Perform complex cryptographic operations
  - Store and manipulate data in non-volatile and volatile memory

The terms “Java Virtual Machine” and “JVM” mean a Virtual Machine for the Java™ platform.

Java Platforms



Optional Packages

Java Platform, Enterprise Edition (Java EE)

Optional Packages

Java Platform, Standard Edition (Java SE)

Personal Basis Profile

Personal Profile

Foundation Prof.

CDC

Optional Packages

MIDP

CLDC

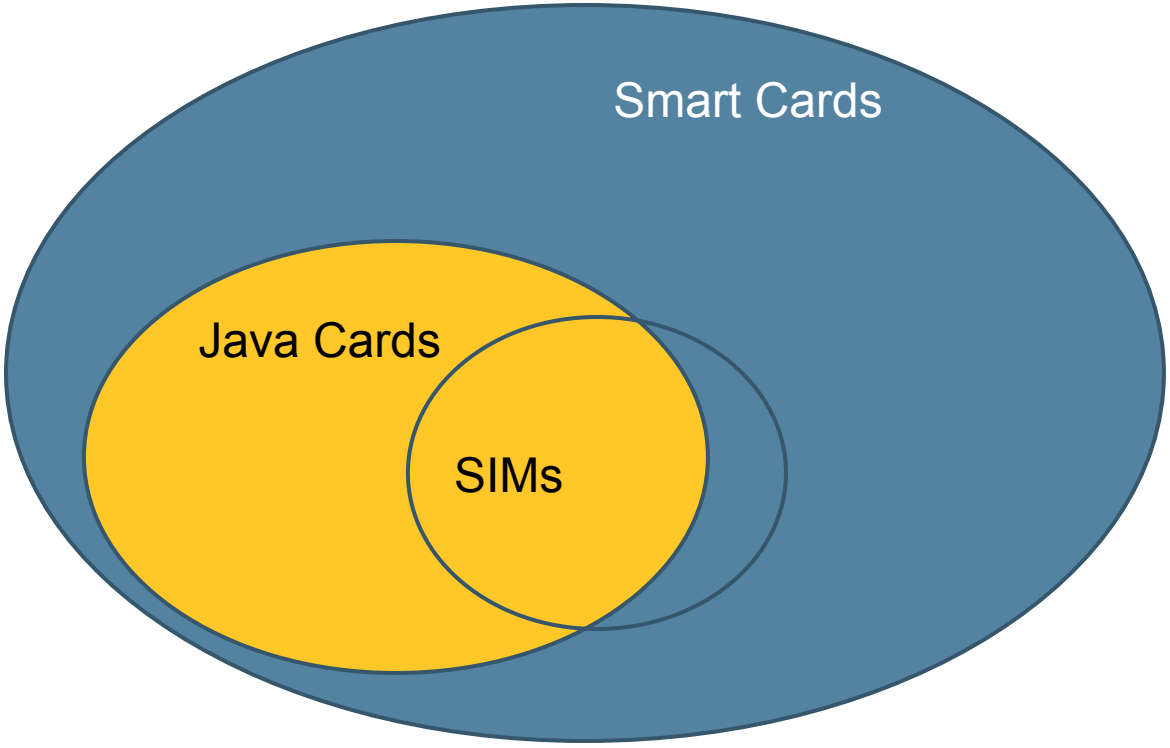
Java Card API

JVM machine

KVM

Card VM

Smart, Java Platform, and SIM Cards



Communication With Smart Cards

- Strongly standardized by international and industry specific organizations
  - ISO (e.g., ISO 7816), ETSI
- Master-slave based
  - Terminal is the master and initiates the communications
  - Terminal and cards negotiate a protocol speed
- APDU—Basic app. layer protocol
  - Application Protocol Data Unit
  - 256 bytes in 255 bytes out

ISO 7816 for Electronic ID Cards

- ISO 7816-4 specifies
  - Command-response pairs
  - Retrieval of data elements in the card
  - Structure for storage in the card
  - Access methods
  - Security architecture
  - Secure messaging
- Does not cover internal implementation
- Independent of physical interface technology
  - Contacts, close coupling, or radio frequency

Agenda

Overview of Smart Cards and SIM

Overview of SATSA API (JSR 177)

Using SATSA-APDU

- Control flow, APDUs, access control

SATSA APDU on Nokia devices

DEMO

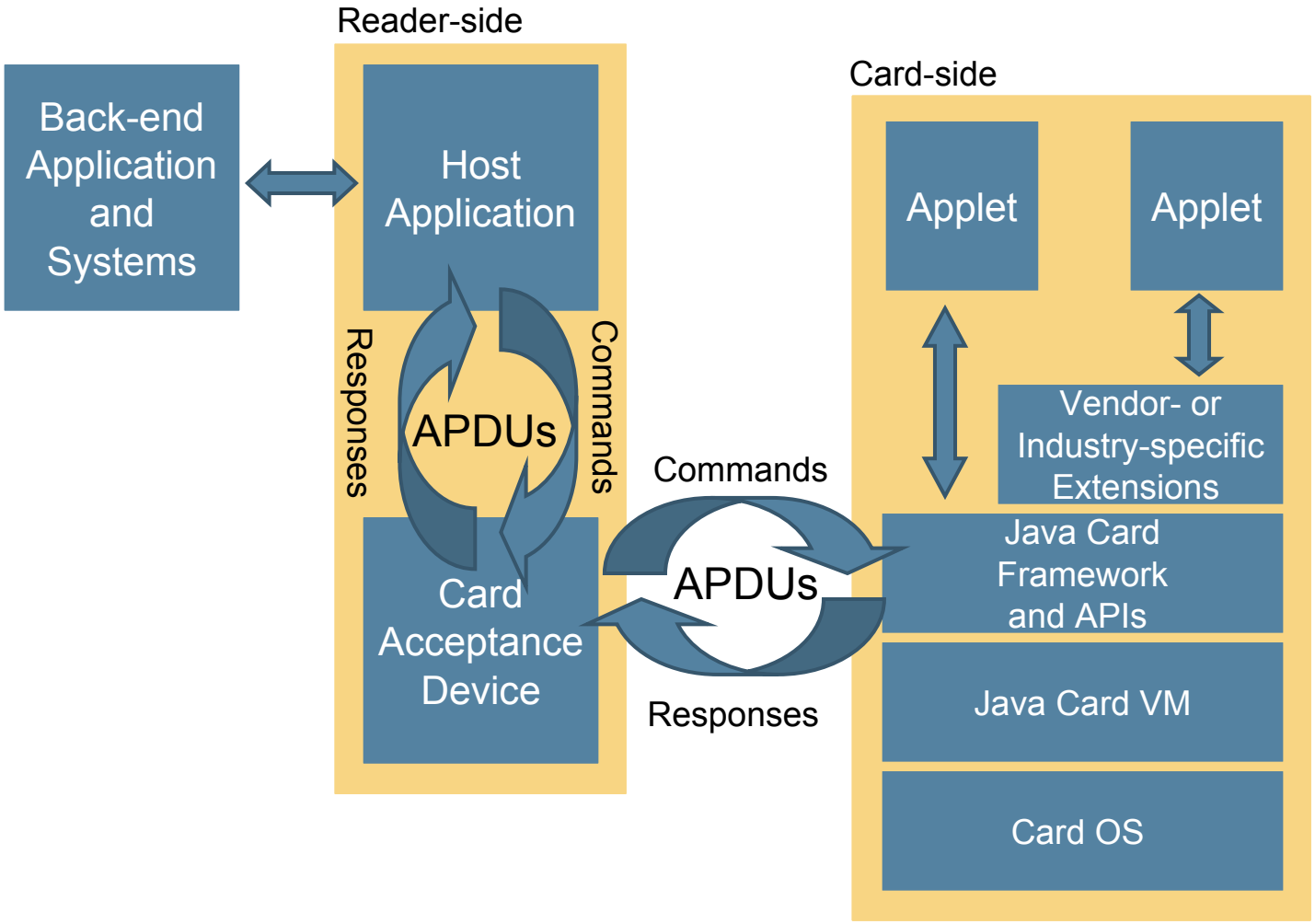
Why an Interface Between Java Platform, Micro Edition (Java ME Platform) and Smart Card?

- Enabling the communication between Java ME platform and Java Card platform
  - The two most widely used and adopted application platform in the wireless space
- Enable a split architecture of the application
  - Use Java ME platform for UI and user interaction, network communication, communication via external interfaces (e.g., IRDA, Bluetooth)
  - Use the card to create, store, and process user or service provider credential, settings, data, all kind of data that should be easily transferred from one terminal to another one

Example Use Cases

- UI for SIM-Toolkit application
  - e.g., mobile payment
- Secure token for existing MIDlet, MIDlet acting as a proxy between the card and a service online
  - Secure Web services
  - Enterprise connectivity
- Storing user and UI settings on the card
- Authentication, signatures on the card
  - Trusted personal device

Smart Card Application Architecture



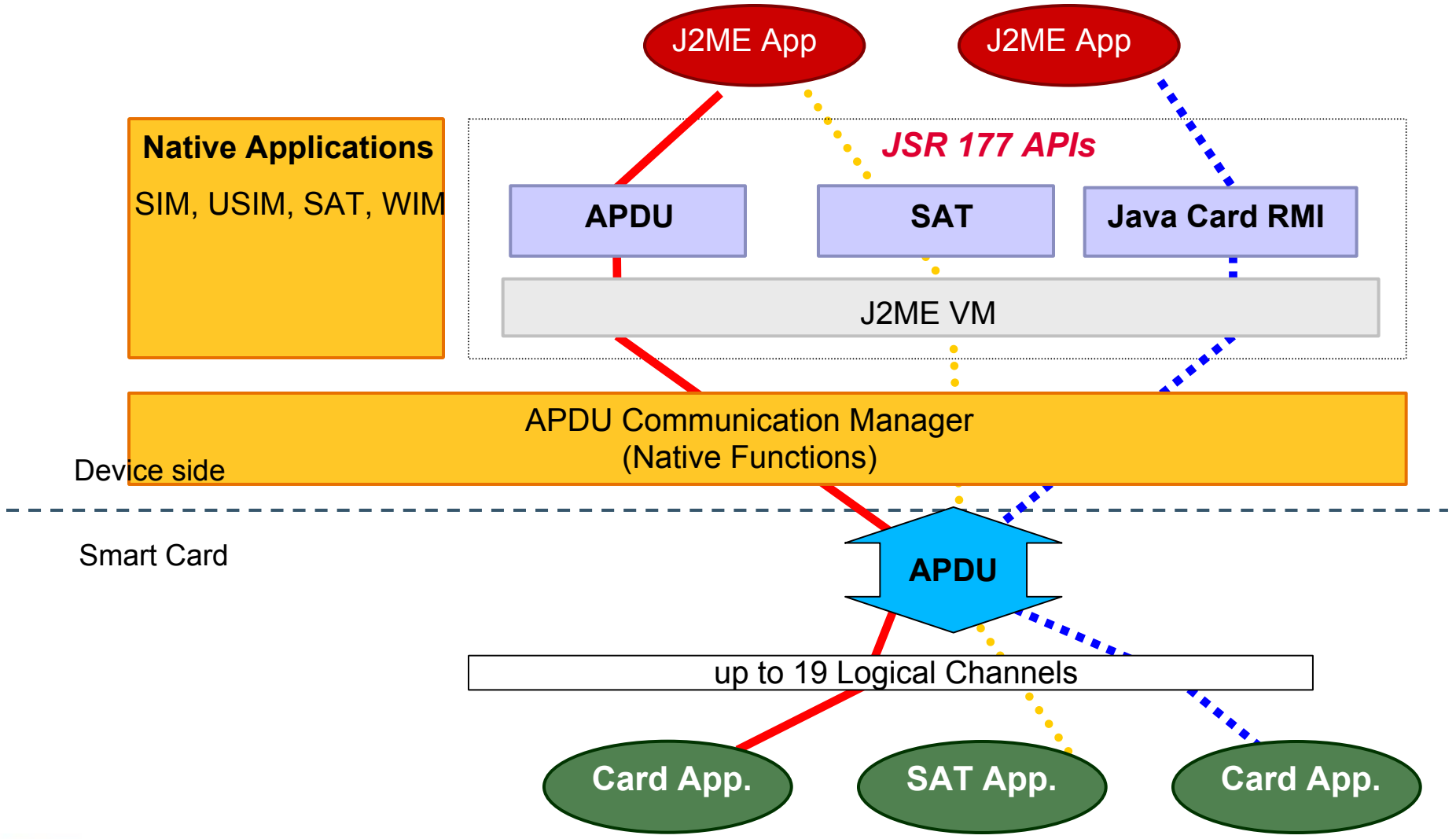
Design Criteria

- Small, easy to use
- Focusing on basic smart card functionality
- No high-level framework
- Protecting the card against attacks and misuse
  - Note Appendix A in API specification
- Support for SIM Application Toolkit (SAT)
  - Migration path for older cards with no logical channels
- Covers not only the SIM card but general smart card use
  - e.g., second slot in the terminal, POS device

Security and Trust Services API (JSR 177)

- Specification finalized September 2004
  - Spec lead: Sun Microsystems, Inc.
- Consists of four optional packages (can be implemented independently)
  - SATSA-APDU—communicating with smart cards
  - SATSA-CRYPTO—encryption (subset of Java Cryptography Extension (JCE))
  - SATSA-JCRMI
  - SATSA-PKI—digital signatures and user credentials

High-Level Architecture



SATSA-APDU

- Communication with smart cards
  - ISO7816-4 compliant
- Uses APDU protocol
  - APDUs are implemented by ALL card deployed in the world today
- Based on Generic Connection Framework (GCF)
- Two components
  - `APDUConnection (javax.microedition.apdu)`
  - `UnsupportOperationException (java.lang)`

JSR 177 and Mobile Services Architecture (JSR 248)

- JSR 248—removing fragmentation
  - Two sets of APIs
    - MSA Subset of 8 APIs and MSA of 16 APIs
  - Additional clarifications
- JSR 177 is part of the MSA set of APIs
  - APDU—conditionally mandatory
  - CRYPTO—mandatory
  - Sun™ PKI hardware—conditionally mandatory
  - Java Card Remote Method Invocation—not part of MSA

APDU Clarifications in JSR 248

- Access Control mechanisms must be implemented
- Security permissions
- System property: `microedition.satsa.apdu.version`

JSR-257 Contactless Communication

- `ISO14443Connection`
  - Does not replace `APDUConnection`
  - May take advantage of JSR 177
  - Communication using APDUs
- JSR 177 only for resident smart cards
  - Only specific application on the card
- JSR 257 for resident and external smart cards
  - Access to the whole smart card

Agenda

Overview of Smart Cards and SIM

Overview of SATSA API (JSR 177) and related JSRs

Using SATSA-APDU

Control flow, APDUs, access control

SATSA APDU on Nokia devices

DEMO

Using APDU

- Very simple process
  - Open connection
  - Send and receive APDUs
  - Close connection
- The main challenges are in the access control
  - API access rights (requires signing)
  - Allowed APDUs

Typical Use of `APDUConnection`

- Get the connection from Generic Connection Framework
  - Generic connection URI string with a card application identifier (AID)

```
APDUConnection cardConn =
  (APDUConnection)Connector.open("apdu:0;
  target=A0.00.00.00.62.03.01.0C.02.01);
```

- Exchanging APDUs

```
byte[] response =
  cardConn.exchangeAPDU(byte[]);
```

- Closing the connection

```
cardConn.close();
```

Application Protocol Data Units

- Short message represented by bytes

```
private final byte[] sampleAPDU =
  // Select File EF_ID
  {(byte)0x00, (byte)0xA4, (byte)0x08,
  (byte)0x00,
  (byte)0x02, (byte)0x00, (byte)0x03 };
```

- Two types of messages
  - Command APDUs (to smart card)
  - Response APDUs (from smart card)
- Ordinary APDUs vs. SAT APDUs

Connection Object

- The `APDUConnection` Object protects the SIM card
  - Communication is application centric
    - MIDlet to card applications
  - It is not possible to reset the card
    - The getATR() method retrieves a cached Answer to Reset (ATR)
  - It is not possible to SELECT a network access application
    - Booth operations would break the network communication
  - The logical channel is assigned by the card and managed by the runtime environment
    - Not exposed in the application level (to the MIDlet)
      - Prevents that one MIDlet occupies all available communications channel

Access Control for SATSA-APDU

- Access Control Entries (ACEs) in Access Control Lists (ACLs) in a Access Control File (ACF)
  - Published by the smart card itself
- ACE indicates
  - Terminal object (principal)—MIDlet
  - Permissions
  - User authentications
- See Appendix A, Recommended Security Element Access Control in JSR 177 spec

Access Control Details

- Opening an `APDUConnection`
  - ACE principal (domain category) matches MIDlet domain category
  - ACE principal (domain root) matches MIDlet certificate chain
  - ACE principal (end-entity) matches MIDlet signing certificate
- Sending an APDU
  - APDU is specified at least in one ACE
  - Not application selection or channel management APDU

Security Using PINs

- `APDUConnection` has a number of PIN handling methods
- All PIN methods have to invoke the native PIN UI
  - Distinguishable to MIDlet UI
- Note: MIDlet never sees the actual PIN
  - PINs indicated with a pinID

Technical Constraints

- CLA byte `0x0X, 0x8X, 0x9X` or `0xAX` not allowed on connections to logical channel number > 0
- Select and Manage Channel APDU sent by the MIDlet will raise an exception
- No direct access to the file-system of the card
  - Can be solved by a proxy applet in the card

Technical Constraints for SAT

- Available only in operator domain
- Only ENVELOPE APDUs may be sent
  - For all other APDUs IllegalArgumentException is thrown
- The class byte MUST be set by the implementation
  - Either A0 or 80 (GSM or UMTS)
  - The class byte supplied by the Java 2 Platform, Micro Edition (J2ME™ platform) application will be ignored
- When the J2ME platform application sends an ENVELOPE APDU to the SIM, the native application may be performing a proactive session. In this case the SIM MUST manage the synchronization issue. The SIM may respond with status word '93 00' (SIM Application Toolkit is busy) when the SIM is performing another proactive session

Security Permissions for APDU

- Protected API calls
  - Function Group: Smart Card Communication

```
javax.microedition.apdu.aid
Connector.open("apdu:"[<slot>]";target="<AID>);
```

  - SAT communication

```
javax.microedition.apdu.sat
Connector.open("apdu:"[<slot>]";target=SAT");
```

- Untrusted MIDlets—No access
- Function Group: Smart Card Communication
  - Trusted 3rd-party MIDlets
    - Default—session
    - Other settings—Blanket, No
- Operator domain required for SAT

Tools for Creating APDU Applications

- Your favorite Java IDE
- Emulators for testing
  - Sun Java Wireless Toolkit 2.5
    - MSA-compliant
  - Nokia Series 40 3rd Edition Feature Pack 2
    - Supports APDU package of SATSA API
- Smart card reader

Agenda

Overview of Smart Cards and SIM
Overview of SATSA API (JSR 177)
Using SATSA-APDU
  Control flow, APDUs, access control
SATSA APDU on Nokia devices
DEMO

APDU and Nokia

- Nokia Series 40 3rd Edition Feature Pack 2 devices and Series 40 5th Edition platform
  - http://forum.nokia.com/devices
- Included also in the SDK
  - Supports T=0 smart card communication
  - Smart card reader needed with SDK
    - Tested with Gemplus reader
- Note: Not available on S60 devices yet
  - CRYPTO and PKI supported starting from S60 3rd Edition

Nokia Devices With SATSA-APDU

Nokia 3110 Classic

Nokia 5200

Nokia 5300

Nokia 6085
Nokia 6086

Nokia 6300

Nokia 7373

Nokia 7390

Series 40 3rd Edition Feature Pack 2 platform

DEMO

Using SATSA-APDU on a real device

Summary

- SATSA-APDU is used to execute commands on smart cards
- Access to smart card is restricted
- Limited availability on real devices
- Java Wireless Toolkit 2.5 and Nokia Series 40 3rd Ed FP2 and Series 40 5th Edition

For More Information

ISO 7816 standard

http://java.sun.com/products/javacard

http://www.forum.nokia.com

Q&A

Sebastian Hans
Hartti Suomela

What to Do With APDU?

Sebastian Hans

Sr. Tech. Expert
Sun Microsystems, Inc.
http://www.sun.com

Hartti Suomela

Sr. Tech. Expert
Forum Nokia
http://forum.nokia.com

TS-5642