RENESAS DEVCON

Enabling the Smart Society

OCTOBER 22-25, 2012
HYATT REGENCY ORANGE COUNTY

RENESAS

jNet Technology, Inc.

BL01A - Java & Global Platform Applet Development

Mikhail Friedland - jNet Technology, Inc.

Class ID: **BL01A**

Renesas Electronics America Inc.

Mikhail Friedland

- Concise Biography
  - President of jNet Technology since 1998
  - Contributor to early JavaCard and Visa OP implementations at Sun and Visa
  - 15 years in smart card industry
  - Specializing in compact Virtual Machines, cryptography and VM applications in embedded control and factory automation.
  - Previously worked in telecom and paperless medical office industries.

Renesas Technology & Solution Portfolio

Agenda

- JavaCard Architecture Overview
- Introduction to Development Environment
- Managing executable content on JavaCard
- Global Platform Architecture & Internals
- Using Cryptography on a Smart Card
- Summary
- Q & A

jNet Java Card Solutions on Renesas RS47x

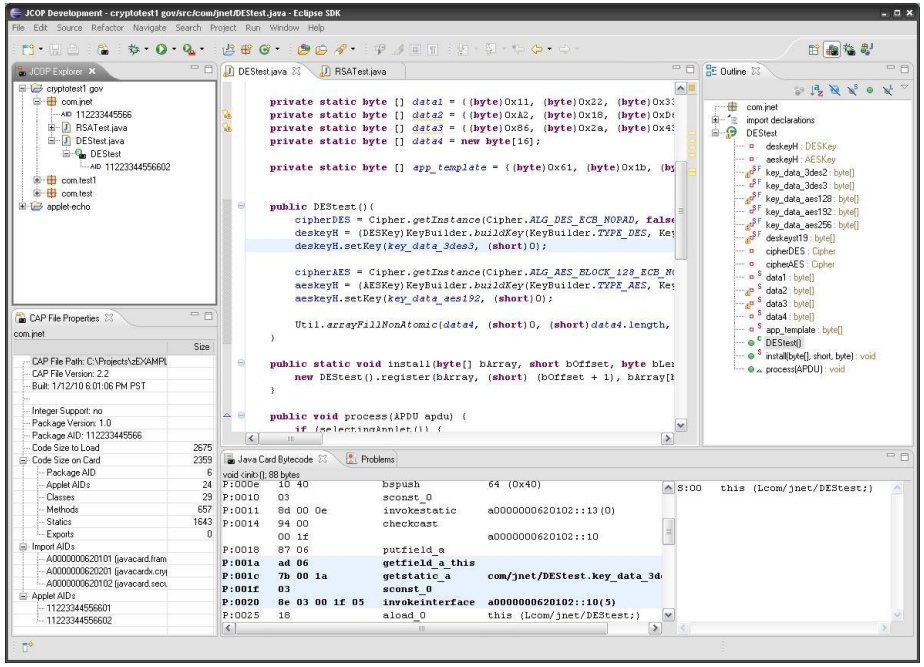
- Modular & Scalable Design
- High Performance Java Execution
- Dual I/O Solutions
- Secure Implementation:
  - FIPS 140-2 Approved Mode of Operation
  - Common Criteria & JavaCard Protection Profile
- Roadmap:
  - Government ID
  - Banking
  - Transit & Loyalty
  - GSM

Java Card Development - Overview

- Applet developer perspective
  - Development environment
    - Eclipse Compatible
    - Global Platform card edge commands (shell based approach)
  - Design techniques for Java Card applets
  - Loading Java Card applets
  - Working with APDUs / Shell
  - Debugging applets
  - Advantages & Limitations

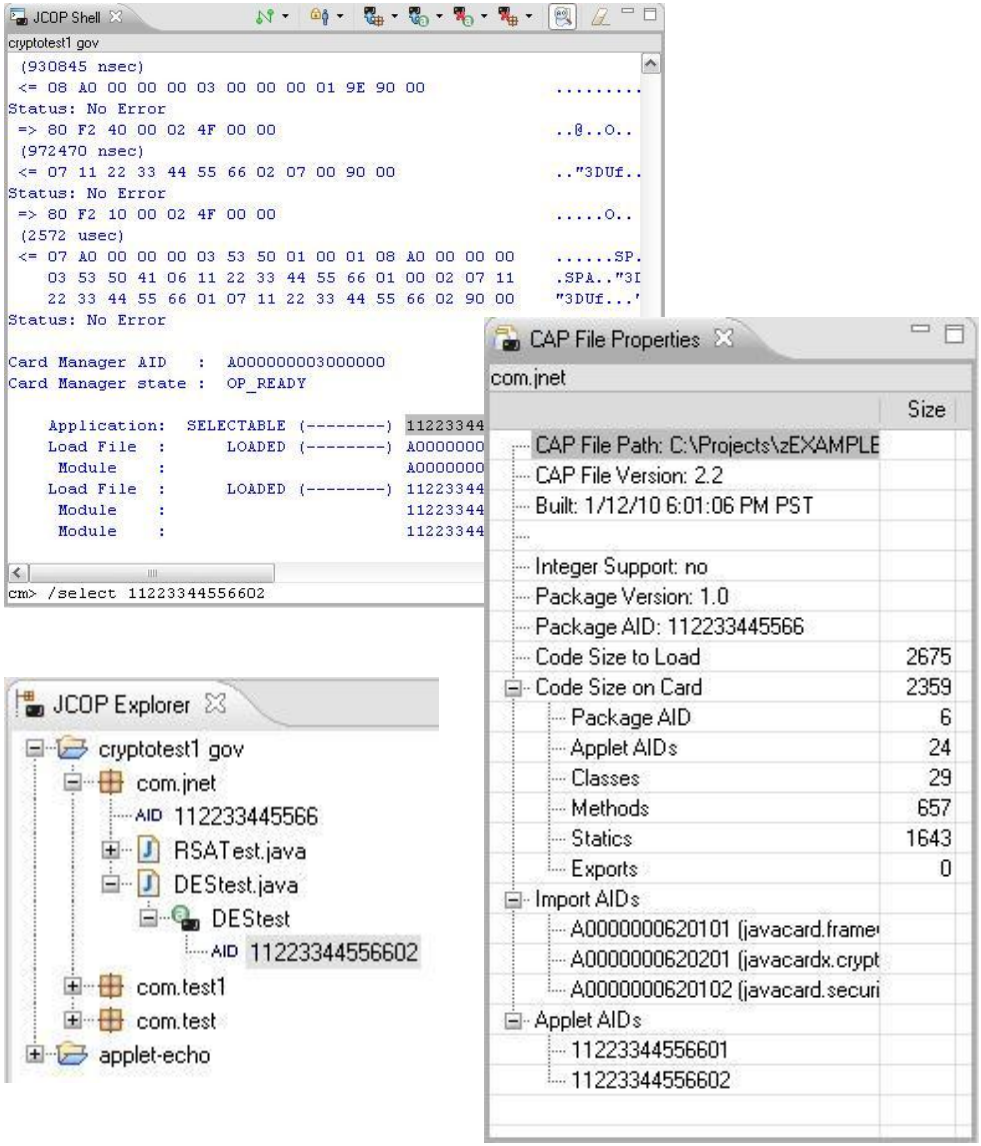
Development Environment

- Eclipse IDE + jNet tools plug-in
  - Specific JavaCard Views
  - Target device
    - Virtual Card Simulator on Win32 (jNet)
    - Real JavaCard (Renesas)



Development Environment

- Eclipse Shell
- Eclipse Explorer
- CAP File properties



Development Environment

- Java Card memory
- Java Card bytecodes

Java Card Memory

cryptotest1 gov

Property	Value
Free EEPROM	75096
Largest free EEPROM block	75096
Used EEPROM	3769
Free Stack	222
Used Stack	34
Free Transient Heap	2180
Used Transient Heap	716

Java Card Bytecode

void process(APDU); 185 bytes

```
P:0000   18          aload_0          this (Lcom/j
P:0001   8b 00 1b    invokevirtual    a00000006201
P:0004   60 1c       ifeq             +28 (P:0020)
P:0006   19          aload_1          apdu (Ljavac
P:0007   8b 00 26    invokevirtual    a00000006201
P:000a   3b          pop
P:000b   19          aload_1          apdu (Ljavac
P:000c   7b 00 2b    getstatic_a      com/jnet/DES
P:000f   92          arraylength
P:0010   8b 00 09    invokevirtual    a00000006201
P:0013   19          aload_1          apdu (Ljavac
P:0014   7b 00 2b    getstatic_a      com/jnet/DES
```

```
S:00   this (Lcom/jnet/DEStest
S:01   apdu (Ljavacard/framewc
S:02
S:03
S:04
```

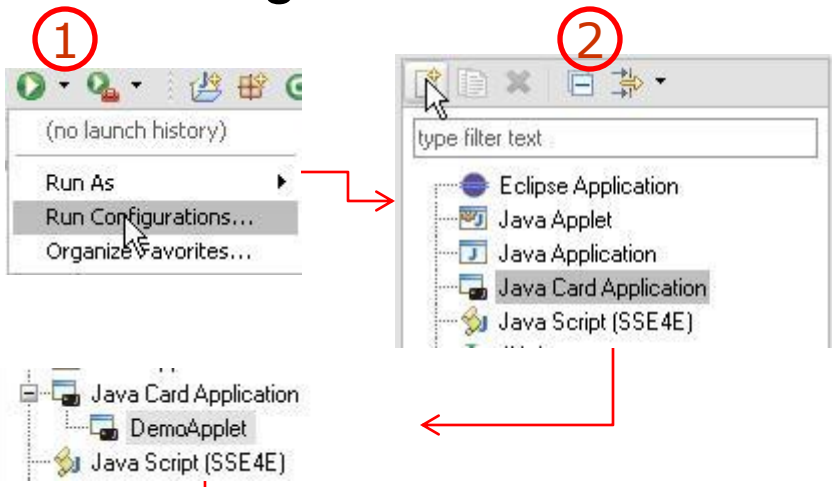
Design Techniques with SmartCard on Eclipse

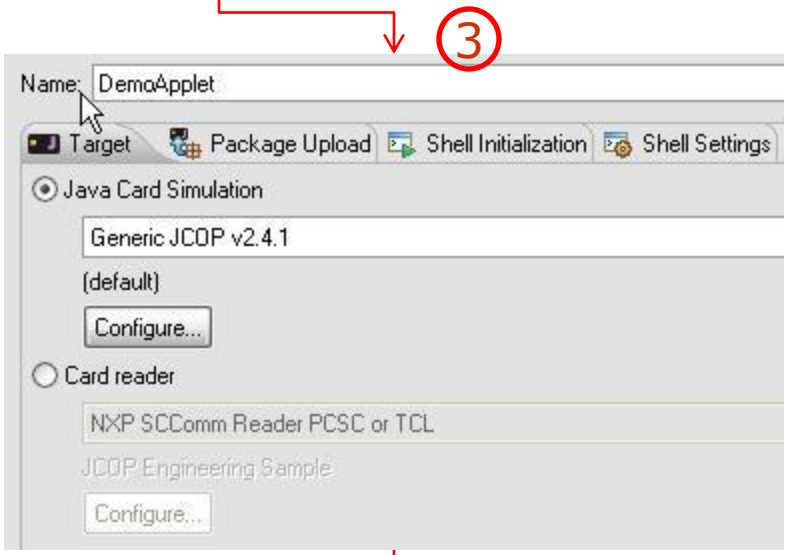
- Main Applet
  - Extends JavaCard applet class
  - Process method handles APDUs
  - Dispatches to function depending on INS
- Watch the bytecodes
- No static vars pointing to another applet
- Keep things simple
- Keep the application in one package

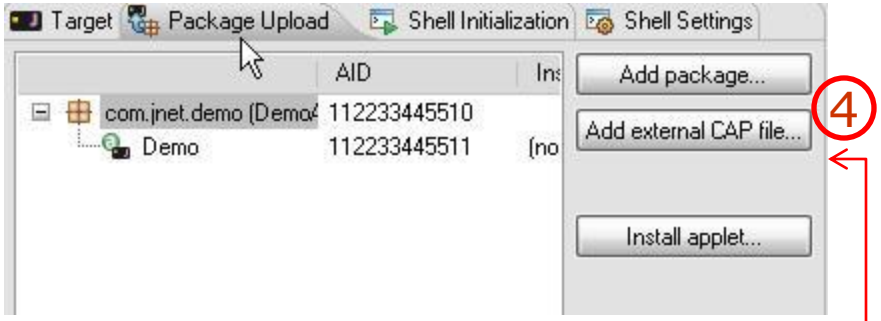
Loading Applets

Create & Configure a run configuration

1. Select your java card project and click "run Configuration" from the run menu
2. Select "Java Card Application" and click "new"
3. Configure target device
4. Select packages to load & applets to install

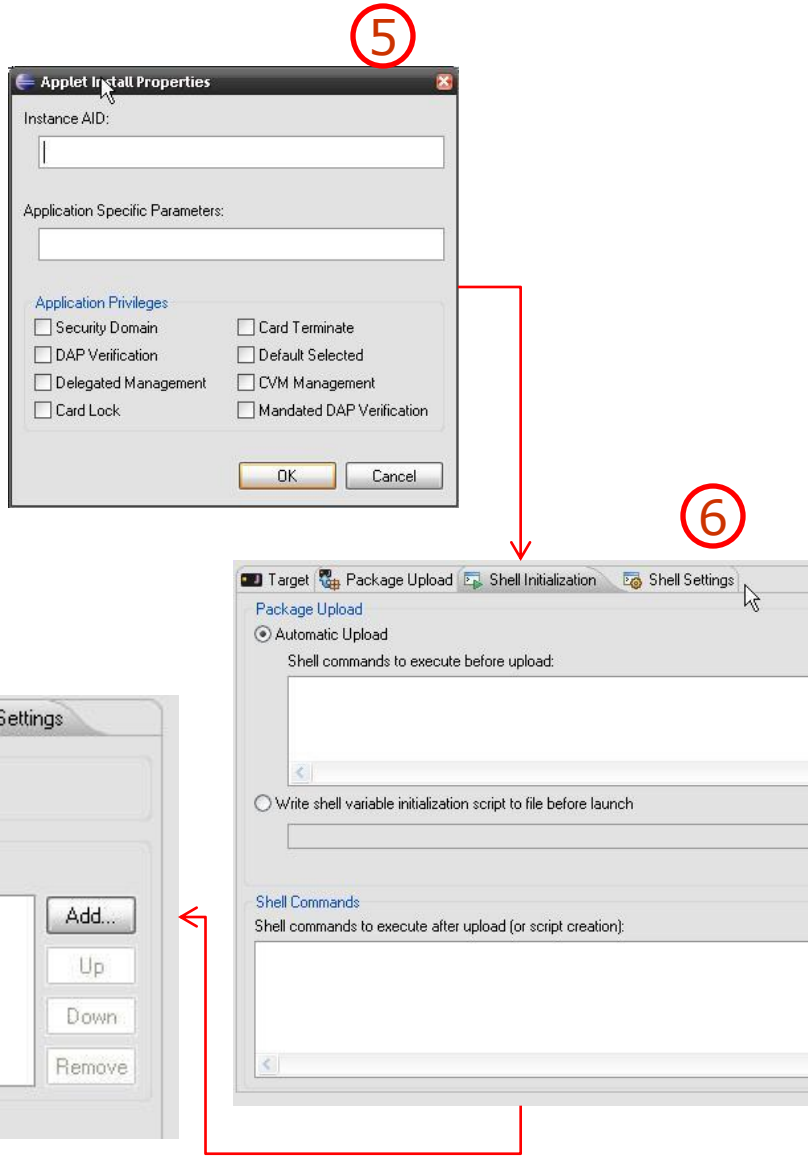






Loading Applets

5. Set installation parameters and privileges of the applet
6. Set before and after upload scripts (optional)
7. Set script folder (optional)
8. Run configuration



Working with APDUs / Shell on Eclipse

- Basic commands
  - card-info
  - /atr
  - /select AID
  - /send CLAINSP1P2LC
  - /close
  - help
- Using scripts
  - .jsch file
  - Setting script folder
  - Execute a script by typing its name in the command shell

Debugging Applets with Eclipse IDE

- Debugging like regular java in a simulator
  - Set breakpoints
  - Step through the code
  - Watch variables
- Use the shell to send APDU commands
- More details with javacard bytecodes

Java Card Architecture

- Java Card VM
  - Built-in language security
  - Data types
  - Runtime environment specifics
- Main components of Java Card VM
  - Method contexts
  - Runtime structures
  - Stack frames
  - Objects representations
  - Exceptions & error handling

Applet Loading/Install/ Deletion process

- Java Card Applet Execution
  - CAP files
  - Creating applet instance
  - Starting the Virtual Machine Engine
  - Interpreting the opcodes
  - Method calling & return
  - Exiting the applet

ROM Mask structure

- Java Card API sub-system
  - java.lang
  - javacard.framework
  - javacard.security
  - javacardx.crypto
- API implementation and native linkages
- Native OS code
- Tools for developing custom packages

Building Applets

- Export files – internal details
  - Internals of Java linkages
  - Constant pool entries
  - Classes & interfaces exposed
  - Fields & methods exposed
  - Attributes
  - Hierarchies

CAP Files

- CAP File internals
  - Directory structure & component model
  - Installation sequence
  - CAP file components
    - Header & directory
    - Applet
    - Import
    - Class, method, static field
    - Reference location, export, descriptor

Java Card Architecture

- VM Opcodes
  - Why 8-bit bytecodes?
  - Required Java bytecodes for JCVM
  - Reserved opcodes
  - Optional opcodes
  - Runtime error handling & security exceptions
  - Instruction set – brief overview

Memory management

- Memory Types on Card
  - EEPROM memory management
    - Persistent storage
  - RAM memory management
    - Transient Arrays:
      - Clear on Reset (COR)
      - Clear on Deselect (COD)
    - Java stack
      - Temporary storage within method context

HAL: RS4x Family Specifics

- RS4x family as it relates to JavaCard
  - Mapping RS4x internal architecture to 16-bit VMs
- Address spaces
- Portability Issues between Renesas chips
- Optimization

Applet Loading/Install /Deletion process

- Applet Lifetime
  - Install Method
  - Select Method
  - De-Select Method
  - Process Method
  - Register Method
  - Power loss & reset
- Default Applets

Java Card Runtime

- Firewall
  - Applet isolation & object sharing
  - Contexts
- Transactions & Atomic Operations
- Exception handling within JCRE
- APDU class implementation
- Security & Crypto Sub-systems
- JCSystem class implementation

Java Card Runtime

- Applet Installation
  - Resource allocation
  - Registration with JCRE
  - Failures during installation

Java Card Runtime
Atomic Transactions

- Atomic Transaction Mechanism
  - Implementation and Memory allocation
  - Verification of atomic entries
  - Optimization techniques
    - Architecture specific
    - Pre-erasing Eeprom
- Commands processing
  - GP system
  - User applets

Java Card V3.0.1 Advantages

- Advanced architecture
- End-point design – Classic vs. Connected
- Mandatory and optional features
  - Integer types
  - javacardx packages
  - Biometry integration
  - ECC support
  - FIPS 140-2 approved mode of operation
- Support for GP2.2,
- Contactless I/O, TLV, transient asymmetric keys
- More robust test suite by Oracle

JC V3.0.1 Architecture

- Enhanced I/O
  - Logical channel support
  - Contactless Protocols
  - APDU Forwarding
  - Extended APDU Interface
  - Exception handling

Supplementary Logical Channels

- Up to 20 logical channels support
  - Full compliance with JCRE v3.0.1 spec
  - SELECT FILE/MANAGE CHANNEL commands are covered
  - Channels are allocated by blocks of 4 channels at time for better RAM utilization
- VGP211 Limitation of 4 channels
  - Dynamic configuration switch

JC V3.0.1 Architecture

- Extension Packages
  - Math
    - BCDUtil
    - BigNumber
    - ParityBit
  - TLV Processing
  - Util
    - Array logic
    - Integer
    - UtilException

JC V3.0.1 Architecture

- Extension Packages
  - Biometric Extensions
    - Match-on-chip library
    - Native calls
  - Java Card Forum
  - External Memory Interface
    - Mifare I/F

JC V3.0.1 Architecture

- Crypto Enhancements
  - SHA-2 hash suite (SHA-224/256
  - InitMessageDigest
  - Korean SEED (optional)
- Extended JCAPIs
- Easier mapping with GP2.2 features

Crypto Implementation

- Java Crypto APIs
- Pulling parameters off the stack
- Links to native methods
- Keys protection
- Countermeasures

Key Management

- Building keys on-card
- Allocating key objects in Eeprom & RAM
- Protecting keys
  - Static keys
  - Session keys
- Verification of keys prior their use
  - DES & AES (Symmetric keys)
  - RSA (Asymmetric keys)

Crypto Algorithms

- DES & AES
- SHA-1 and SHA-256
- Older hash methods (MD5 & RIPEMD160)
- RSA
- ECC (new ROM mask, Government ID)

GP2.2 Framework - I

- Overview
  - Differences with Java Card specs
  - Card Preparation & Personalization
  - Card Manager
  - Key Usage

GP 2.2 Framework - II

- Security Domains
- APDU Commands
- Open Platform APIs
- Integration with Java Card VM

GP 2.2 Framework - III

- Card Manager
  - Represents Issuer Security Policy
  - Lifecycle States
    - Package
    - Applets
  - Card Content Management
  - Secure Channel Implementation

GP 2.2 Framework - IV

- Card Manager
  - Global PIN
  - Application Locking
  - Card Locking
  - Card Termination

GP 2.2 Framework - IV

- Security Domains
  - Life Cycles
  - Application Access to SD
  - Secure Communication
  - Personalization
  - DAP Verification (PK – DAP)

GP 2.2 Framework - VI

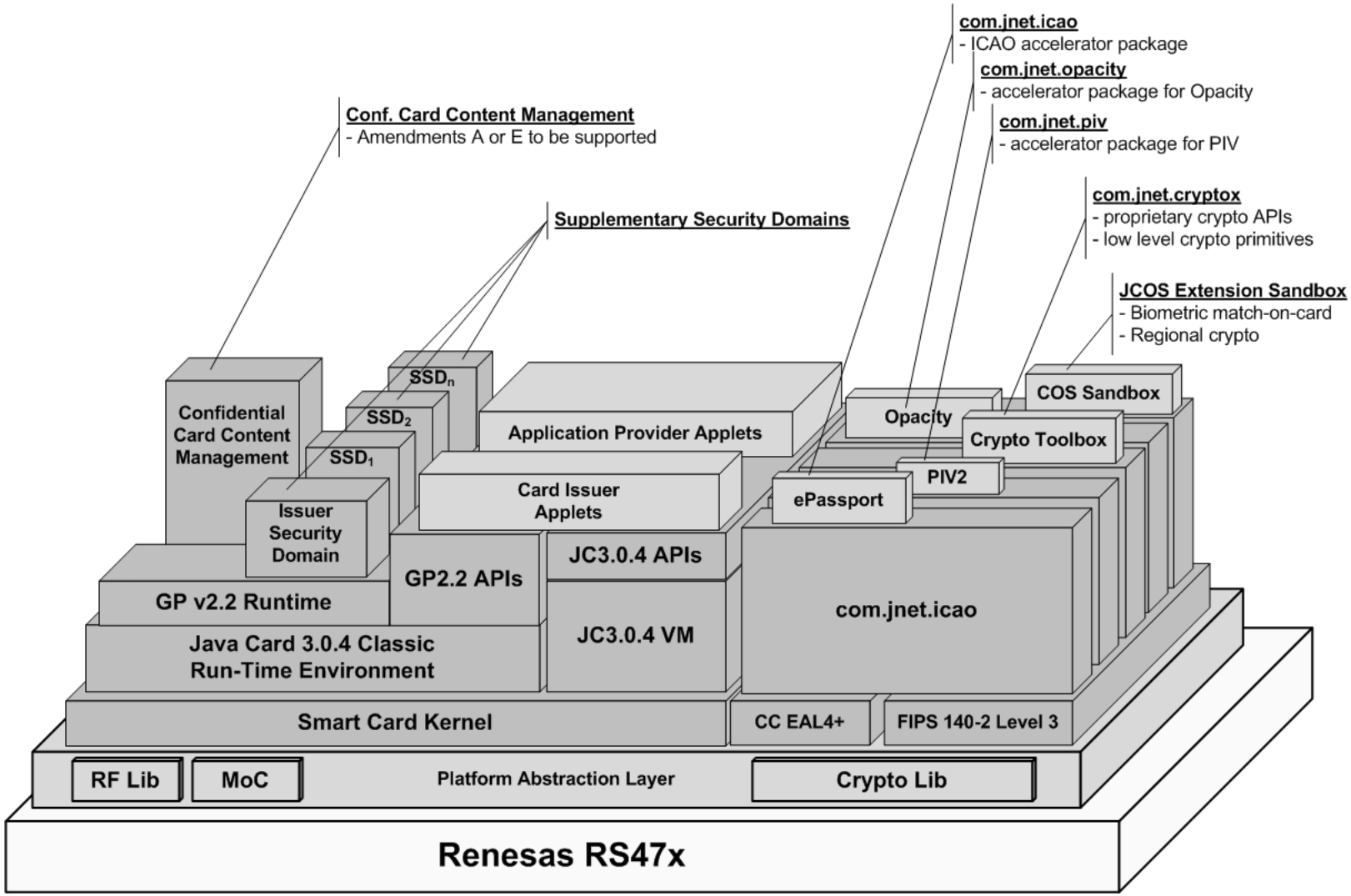
- Global Platform APIs
- APDU Commands
  - GET STATUS
  - GET DATA, PUT DATA
  - INSTALL
  - LOAD
  - PUT KEY
  - SELECT

GP 2.2 Framework - VII

- Miscellaneous Topics
  - Session Keys
  - Hash Usage
  - Authentication Cryptograms
  - APDU Generation & Verification

- What's next for Applet Developers?

jNet Roadmap on Renesas RS47X



DEVCON

Enabling the Smart Society

jNet Technology, Inc.

RENESAS

Renesas Electronics America Inc.