Oracle
Groundbreakers

Java Card

Introducing Java Card Device IO to secure peripherals on I3C bus

Patrick Van Haver
Security Solutions Architect
Oracle - Java Platform Group

Dec. 2021

ORACLE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Java Card

Reference runtime for Secure Elements

Identification

Biometry

Data Confidentiality

Authorization

Secure communication

Access Control

Root of trust

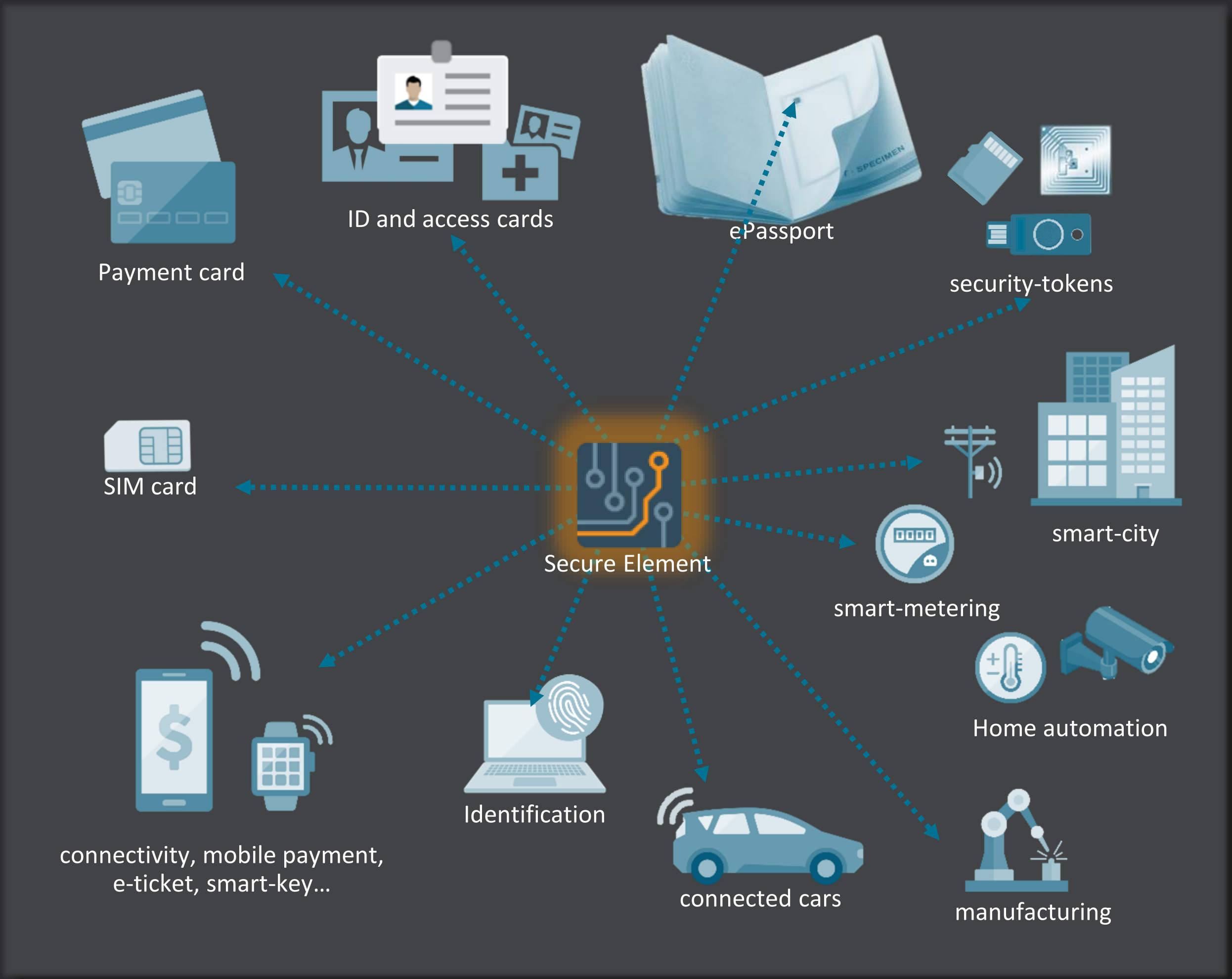
Data Integrity

Secure storage

Secure transactions

Authentication

Device Attestation



Java Card

Reference runtime for Secure Elements

Identification

Biometry

Data Confidentiality

Authorization

Secure communication

Access Control

Root of trust

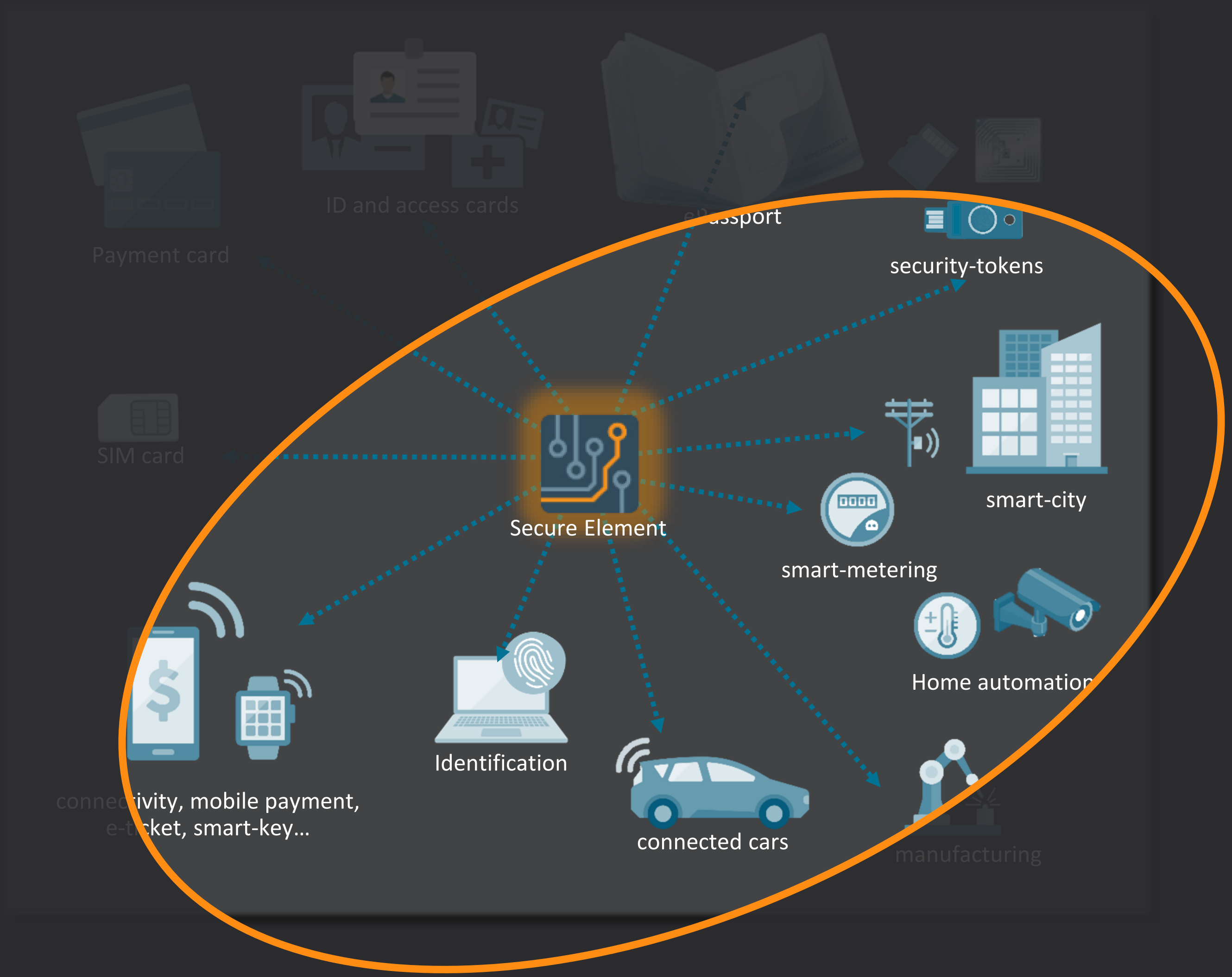
Data Integrity

Secure storage

Secure transactions

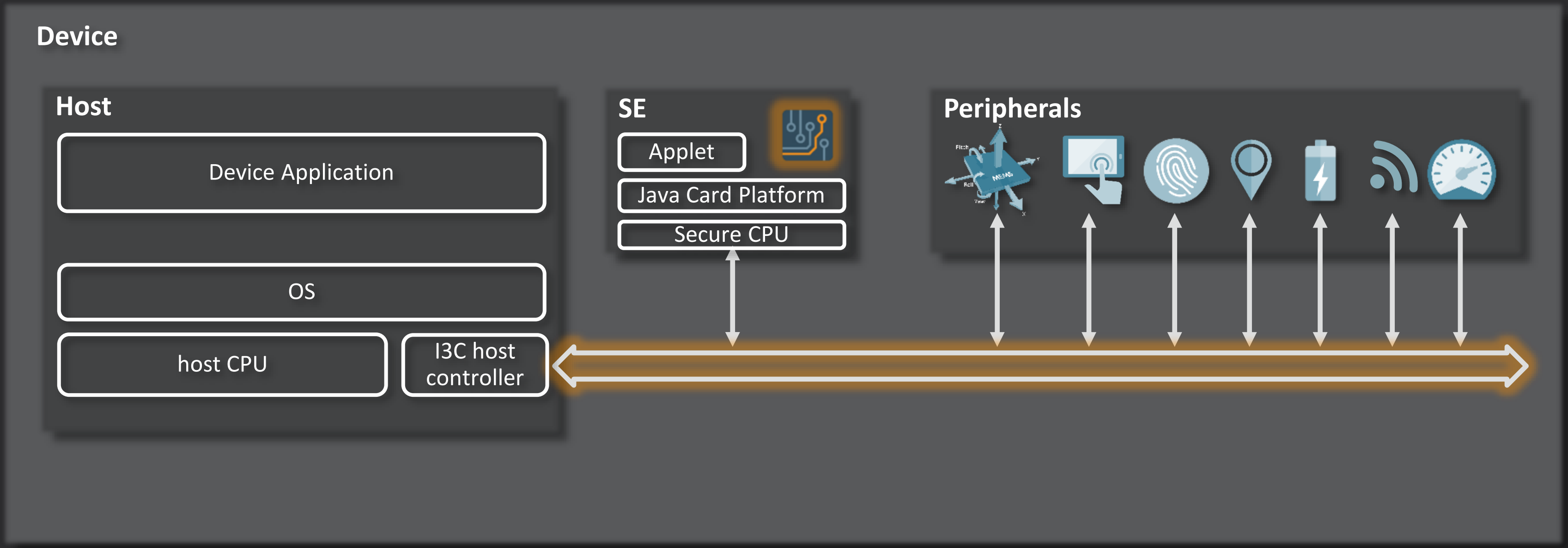
Authentication

Device Attestation



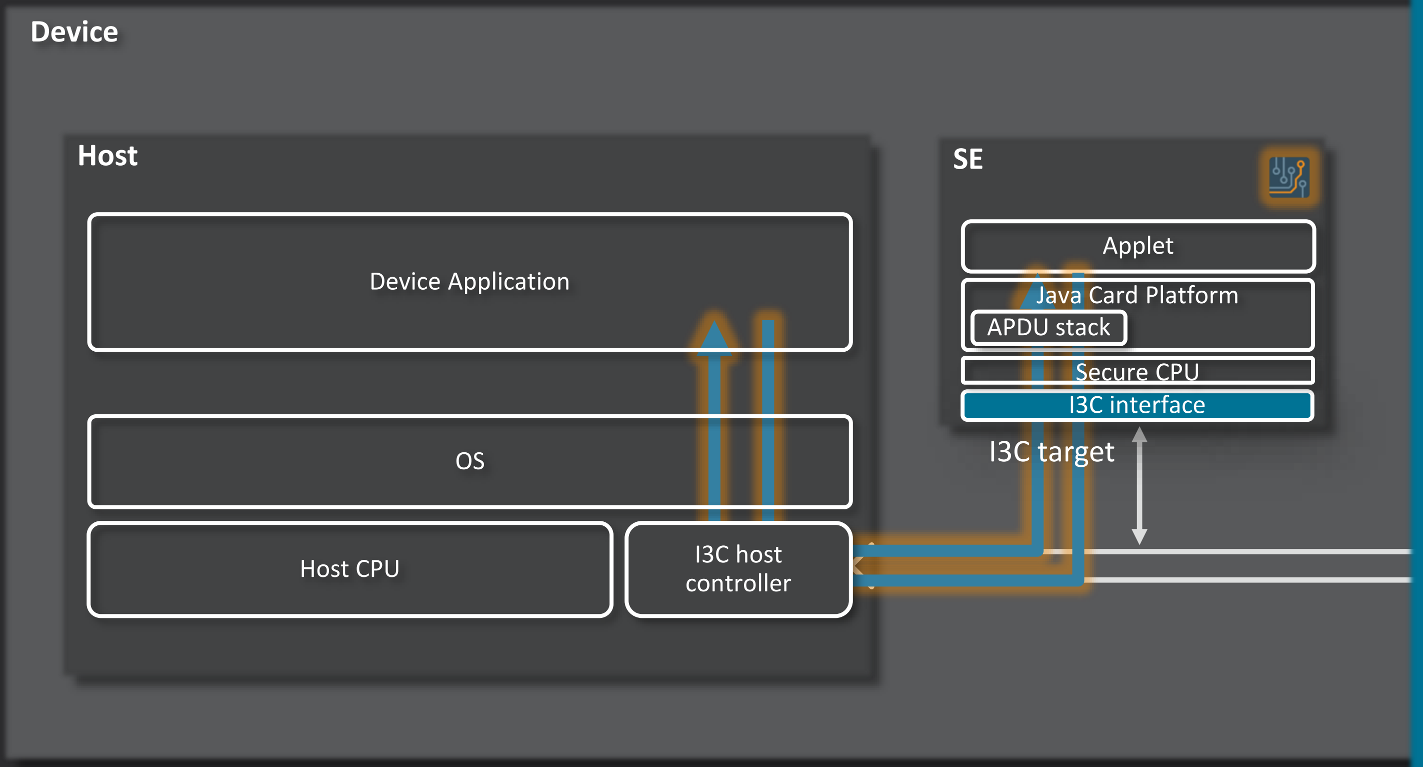
Integration of the Secure Element on the I/O bus

Example using I3C bus



I3C used as the main interface with host

To support existing SE use-cases over I3C I/O interface

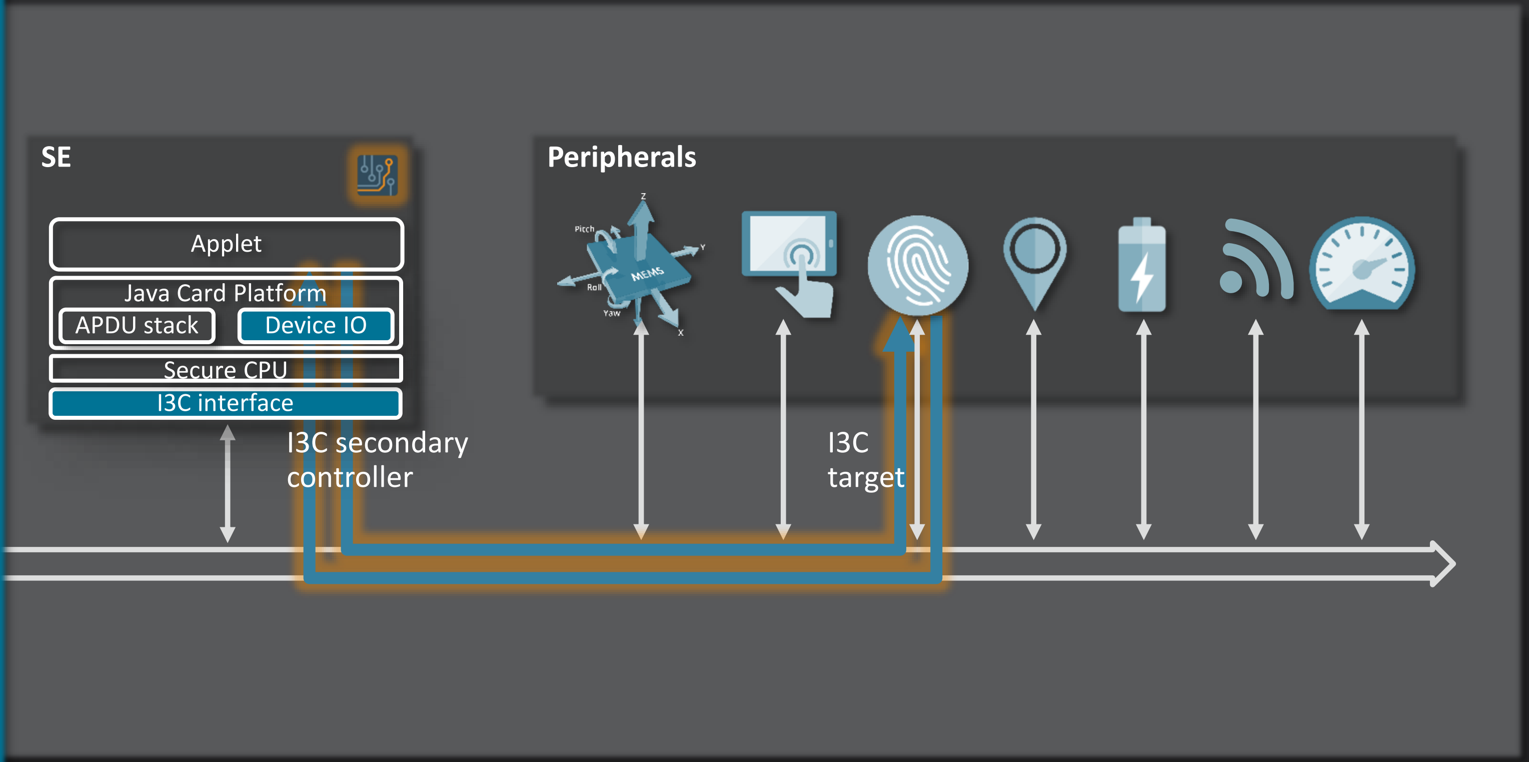


- The SE is an I3C target
  - Replies to requests received from host
- Benefit from I3C interface
  - Simplified integration and wiring with host
  - Lower power consumption
  - Higher data rates
- No impacts on existing applications
  - Only physical layer up to transport layer are updated
  - Encapsulated in APDU class

I3C interface used by Applets to access peripherals

To implement new use-cases using shared peripherals

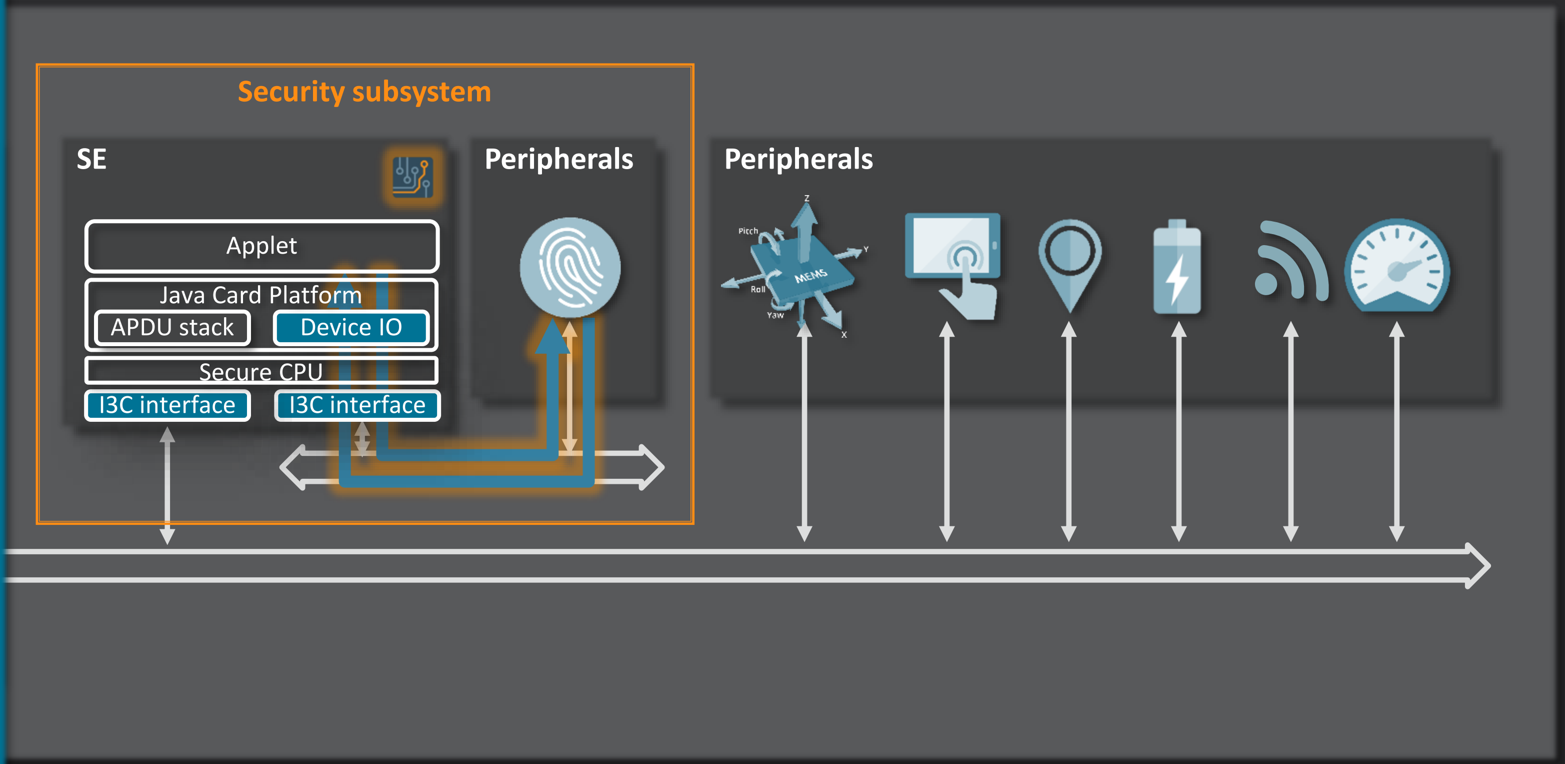
- The SE becomes (secondary) controller
  - Accessing peripherals (I3C targets)
- SE takes control over peripherals
  - To implement new scenarios: configure sensors, read or write data
  - To augment existing use-cases with information coming from sensors (e.g. location info for authentication)
- Requires API extensions
  - Device IO API



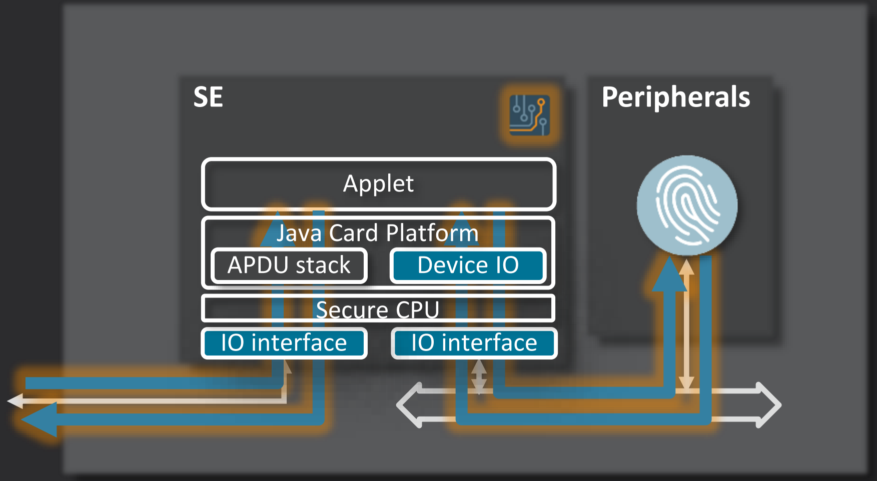
I3C interface used by Applets to access peripherals

To implement new use-cases using dedicated peripherals

- Security Subsystem
  - To enhance security – exclusive control and access to a peripheral
  - To lower power consumption – segmented subsystems (with Hot-Join) independently powered only when needed

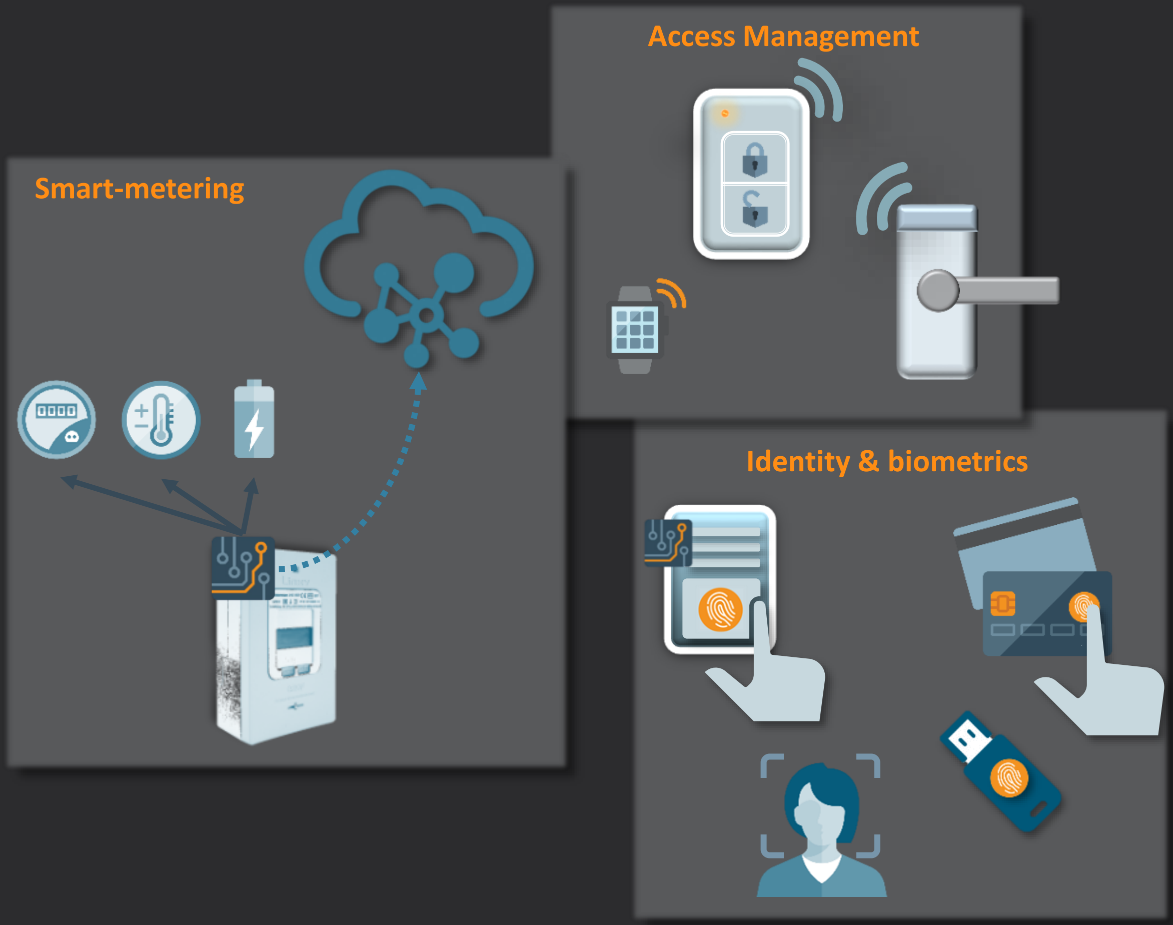


Design also relevant for standalone secure elements

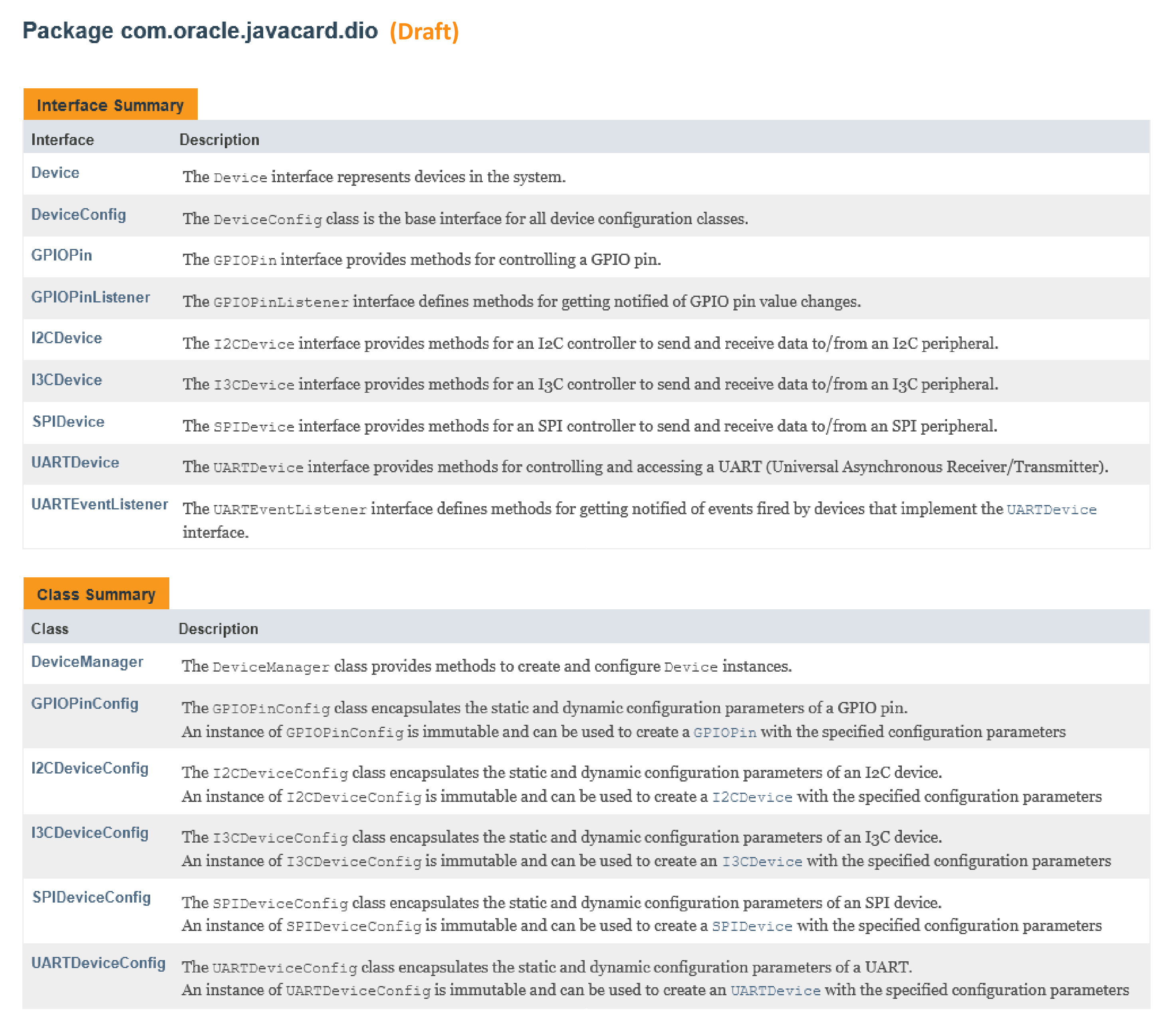


Secure peripherals use-cases

- **Smart-metering**
  Java Card application uses some sensors to detect tampering and access meter data to enforce measurement integrity.
- **Access management (smart-lock, car key, …)**
  Java Card application is triggered when pressing a button and generates a temporary code to open a lock.
- **Biometric identification**
  Java Card application uses biometric sensor to securely capture fingerprint and perform matching
- **Display control**
  Java Card application uses a peripheral to control data to be displayed (dynamic CVV, OTP, crypto-wallet balance, …)



Java Card Device IO API



Package com.oracle.javacard.dio (Draft)

Interface Summary

Interface	Description
Device	The `Device` interface represents devices in the system.
DeviceConfig	The `DeviceConfig` class is the base interface for all device configuration classes.
GPIOPin	The `GPIOPin` interface provides methods for controlling a GPIO pin.
GPIOPinListener	The `GPIOPinListener` interface defines methods for getting notified of GPIO pin value changes.
I2CDevice	The `I2CDevice` interface provides methods for an I2C controller to send and receive data to/from an I2C peripheral.
I3CDevice	The `I3CDevice` interface provides methods for an I3C controller to send and receive data to/from an I3C peripheral.
SPIDevice	The `SPIDevice` interface provides methods for an SPI controller to send and receive data to/from an SPI peripheral.
UARTDevice	The `UARTDevice` interface provides methods for controlling and accessing a UART (Universal Asynchronous Receiver/Transmitter).
UARTEventListener	The `UARTEventListener` interface defines methods for getting notified of events fired by devices that implement the `UARTDevice` interface.

Class Summary

Class	Description
DeviceManager	The `DeviceManager` class provides methods to create and configure `Device` instances.
GPIOPinConfig	The `GPIOPinConfig` class encapsulates the static and dynamic configuration parameters of a GPIO pin. An instance of `GPIOPinConfig` is immutable and can be used to create a `GPIOPin` with the specified configuration parameters
I2CDeviceConfig	The `I2CDeviceConfig` class encapsulates the static and dynamic configuration parameters of an I2C device. An instance of `I2CDeviceConfig` is immutable and can be used to create a `I2CDevice` with the specified configuration parameters
I3CDeviceConfig	The `I3CDeviceConfig` class encapsulates the static and dynamic configuration parameters of an I3C device. An instance of `I3CDeviceConfig` is immutable and can be used to create an `I3CDevice` with the specified configuration parameters
SPIDeviceConfig	The `SPIDeviceConfig` class encapsulates the static and dynamic configuration parameters of an SPI device. An instance of `SPIDeviceConfig` is immutable and can be used to create a `SPIDevice` with the specified configuration parameters
UARTDeviceConfig	The `UARTDeviceConfig` class encapsulates the static and dynamic configuration parameters of a UART. An instance of `UARTDeviceConfig` is immutable and can be used to create an `UARTDevice` with the specified configuration parameters

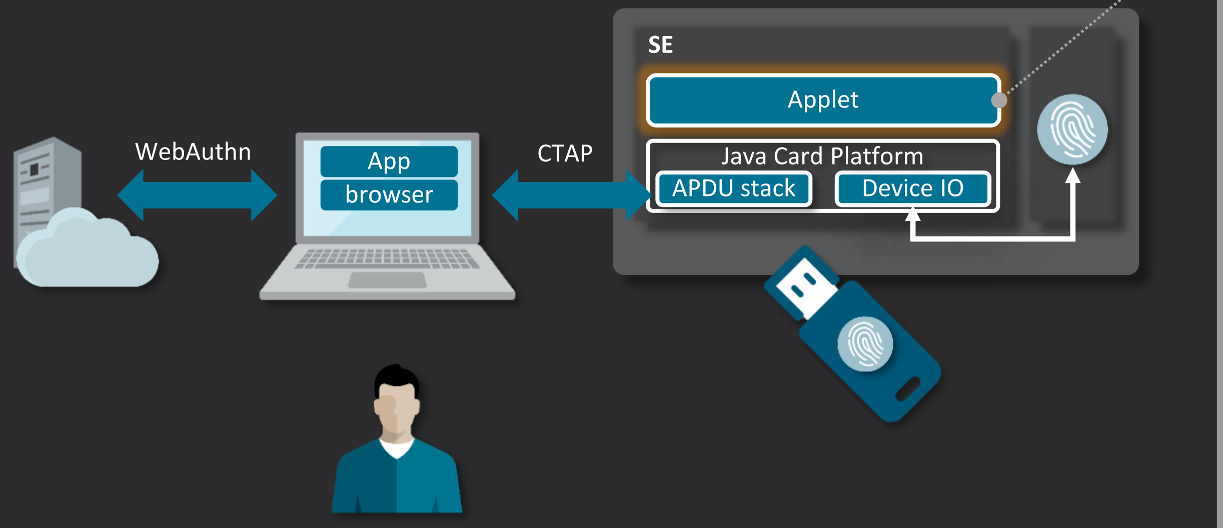
Interfaces to interact with connected devices

- GPIO
- I2C
- I3C
- SPI
- UART

To create and configure Device instances

Configuration parameters for each device type

Example: verifying fingerprint



```
try {
   // get a device instance with specified configuration
   device = DeviceManager.getInstance(IO_ID1,
                I3CDeviceConfig.build(fpaddress, DTM_SDR0));

   // open communication with fingerprint sensor
   device.open();

   // send command to capture fingerprint
   device.write(cmd_get_fp);

   // read fingerprint and check
   len = device.read(rxbuffer);

   res = bioMather.initMatch(rxbuffer.array(), (short)0, len);

   while (res == MATCH_NEED_MORE_DATA) {
      len = device.read(rxbuffer.clear());
      res = bioMather.match(rxbuffer.array(), (short)0, len);
   }
   if ((res >= MINIMUM_SUCCESSFUL_MATCH_SCORE) {
      // success …
   }
} finally {
   // close device when done
   device.close();
}
```

Conclusion

Takeways

- Secure Elements integration into devices requires new IO interface types
- I2C and SPI are already used for the main IO interface of the SE, I3C is on the way
- Secure Elements applications can also access peripherals connected on the same bus, addressing new use-cases and offering effective solutions to enhance device security
- Device IO API is a mean for Java Card applications to access peripherals connected to the secure element and achieve higher security

Questions ?

More Information

https://www.oracle.com/java/technologies/java-card-tech.html

Java Card 3.1 Documentation
Includes all documentation for the Java Card platform and Development Kit.

Java Card Platform Specification 3.1
Latest release of the Java Card specification and the reference for Java Card products.

Java Card Development Kit Tools
The Java Card Development Kit Tools are used to convert and verify Java Card applications. The Tools can be used with products based on version 3.1, 3.0.5 and 3.0.4 of the Java Card Specifications.

Java Card Development Kit Simulator
The Java Card Development Kit Simulator includes a simulation component and Eclipse plug-in.
Combined with the Java Card Development Kit Tools, it provides a complete, stand-alone development environment.

contact: patrick.van.haver at oracle.com