Release Notes

Java Card 2.1 Application Programming Interface Specification

February 24, 1999
Final Revision 1.0

Sun Microsystems, Inc.

901 San Antonio Road

Palo Alto, CA 94043 USA

650 960-1300 fax 650 969-9131

Table of Contents

1. Introduction

This document is the Release Notes for the *Java Card 2.1 Application Programming Interfaces Specification* also known as JCAPI21_FINAL.

Java Card 2.1 Application Programming Interfaces Specification

The *Java Card 2.1 Application Programming Interfaces Specification* describes the programming environment within Java Card technology. It is intended to be used as the foundation for designing implementations and applets for the Java Card technology.

2. Java Card 2.1 Application Programming Interfaces Specification

Major Design Changes in the 2.1 release

This section details the significant changes to the design of the API for the 2.1 release.

New Transience model and API

The Java Card 2.1 API restricts transient objects to arrays of primitive component types and arrays of `Object`. The `System.makeTransient` method has been replaced by the following factory methods in the `javacard.framework` package :

```
JCSystem.makeTransientBooleanArray,
JCSystem.makeTransientByteArray,
JCSystem.makeTransientShortArray,
JCSystem.makeTransientObjectArray.
```

Applet firewall and object-sharing model redesigned

The `System.share` methods in the `javacard.framework` package have been deleted in the Java Card 2.1 API. The applet firewall is now more restrictive and there is a well-defined Shareable interface for object sharing.

Applet `install` method is interoperable

The Applet `install` method in the `javacard.framework.Applet` class uses a byte array as an input parameter instead of the APDU object to allow independence between the installation APDU and protocol, and applet initialization.

New Exception Class hierarchy

The Java Card 2.1 API redefines the `Throwable`, `Exception`, and `RuntimeException` classes in the `java.lang` package to be strict subsets of the standard Java `java.lang` versions. Two new exception classes have been defined for Java Card technology that contain the `reason` field and the accessor methods `getReason` and `setReason`. These are `javacard.framework.CardRuntimeException`, which derives from `java.lang.RuntimeException` class, and `javacard.framework.CardException`, which derives from `java.lang.Exception`.

Multiple instances of single applet class possible

A new method in the Applet class: `register( byte[], short, short )` of the `javacard.framework` package allows an `Applet` instance to provide the AID for applet selection. This allows a single applet class to be registered as multiple applet instances.

AID class is more general purpose

The `AID` class in the `javacard.framework` package is more general purpose and now has a public constructor. It also defines the general purpose `equals` methods to compare `AID` objects. It can be used by applets and API extensions to manage applets and packages.

javacardx.framework (file system) extension package deleted

The `javacardx.framework` package containing the ISO 7816-4 file system classes has been deleted. There has been no consensus on a minimal feature set. Sample applets are able to use internal objects to represent ISO 7816-4 type files and records efficiently.

Cryptography extension packages restructured

The cryptography extension packages have been completely restructured as follows:

The `javacard.security` package is no longer an extension. It contains message digest and signature classes, key interfaces and random number generation classes. This package is suitable for export and does not allow strong encryption.

The `javacardx.crypto` package is an extension package. It contains the cipher class. This class allows encryption and decryption capability and may be subject to export regulations.

The following sections list the changes since the original Java Card API 2.1 release.

API Updates since Java Card API 2.1 Draft 2 Rev 1.4

javacard.framework package changes

Class `APDUException`:

New reason : `T1_IFD_ABORT when the CAD sends an ABORT S-Block command during T=1 protocol inbound or outbound data transfer.`

Documentation Change : BAD_LENGTH reason code now states that setByteLength() may flag this reason if the length is too long in the block chained transfer mode.

Class `APDU`:

New method : `getOutBlockSize()` returns the configured outgoing block size. In T=1 protocol, this corresponds to IFSD (information field size for interface device).

Documentation Change: The method `setOutgoingLength()` method throws `APDUException.BAD_LENGTH` if NO CHAINING transfer is requested and the total transfer length is greater than block size, IFSD.

Documentation Change: The methods `receiveBytes()`, `setIncomingAndReceive()`, `sendBytes()` and `sendBytesLong()` methods throw `APDUException.T1_IFD_ABORT` if the CAD sends an ABORT S-Block command during T=1 data transfer.

Class `OwnerPIN`:

Documentation Change : Added class documentation note about avoiding conditional updates during PIN presentation .

Documentation Change : Clarified the behavior of the `update()` method to state that if a transaction is in progress the pin value and try counter must be conditionally updated.

Documentation Change : Clarified the behavior of the `check()` method to state that even if a transaction is in progress try counter, validated flag and blocking state must not be conditionally updated.

Interface `PIN`:

Documentation Change : Added class documentation note about avoiding conditional updates during PIN presentation

Documentation Change : Clarified the behavior of the `check()` method to state that even if a transaction is in progress try counter, validated flag and blocking state must not be conditionally updated.

Class `JCSystem`:

Documentation Change: Clarified the documentation in the method `isTransient()` method to return `NOT_A_TRANSIENT` when the input object parameter is not an array type.

Class `Util`:

Documentation Change: Documented more precise error checking for the methods : `arrayCopy()`, `arrayCopyNonAtomic()`, `arrayFillNonAtomic()` and `arrayCompare()` to avoid ambiguity under potential exception conditions.

Method redefined : The `arrayFillNonAtomic()` method now returns a short value of `bOff+bLen` instead of void. This makes it consistent with the other `Util` methods.

javacard.security package changes

Interfaces `DESKey, DSAKey, DSAPublicKey, DSAPrivateKey, RSAPrivateKey, RSAPublicKey, RSAPrivateCrtKey` :

Documentation Change : Clarified the description in all the `set..()` methods to indicate that the format of the input data is for plaintext data and input data is copied into an internal representation. Additionally, the `set..()` methods now document that if the input data length is inconsistent with the implementation or key decryption errors occur, a `CryptoException` with reason `ILLEGAL_VALUE` is thrown.

Interface `DESKey` :

Documentation Change : Clarified the description in the `setKey()and getKey()` methods to list the length of the key data in bytes for each of single DES, 2key and 3key triple DES cases.

Class `KeyBuilder` :

Documentation Change : Clarified the description in the `buildKey()`method to state that only keys created by this method may be used with `Signature` and `Cipher`.

Class `Signature` :

Documentation Change : Clarified the description in the `sign()and verify()` methods to state that the state is reset after the signing or verification is done.

Documentation Change : Clarified the descriptions of the DES selectors with 4 byte MAC to state that the MAC value is the 4 most significant byte of the encrypted block.

New algorithm selectors : Added ALG_RSA_SHA_RFC2409 and ALG_RSA_MD5_RFC2409 algorithm options as possible Signature algorithms.

Class `RandomData` :

Documentation Change : Clarified the description in the `getInstance()`method to state that the pseudo random generators seed is initialized to a internal default value.

javacardx.crypto package changes

Class `Cipher` :

Documentation Change : Clarified the description in the `init()`method to state that the key is checked for consistency against implementation only and not against mode. This allows the use of a Private Key in encrypt mode and Public Key in decrypt mode.

Documentation Change : Clarified the description in the `doFinal()`method to state that the state is reset after the encryption or decryption is done.

New Exception possible : Introduced documentation in the `update()` and `doFinal()` methods to state that if the length of the input message is incompatible with the implementation a `CryptoException` with reason `ILLEGAL_USE` is thrown.

Documentation Change : Clarified the description of the ALG_RSA_PKCS1 algorithm option to state that it is intended for short messages.

New algorithm selector : Added ALG_RSA_ISO9796 algorithm option as possible Cipher algorithm choice to be consistent with the Signature algorithm choices. This algorithm is also intended only for short messages.

Interface `KeyEncryption` :

Documentation Change : Clarified the documentation to state that the default decryption `Cipher` object is null and no decryption is performed.

API Updates since Java Card API 2.0

Parameter Checking Policy

A policy for parameter checking is included in the *Java Card 2.1 API Specification*. Some of the benefits of the parameter checking policy are:

- **Predictable Behavior** Conformance to this policy will result in predictable behavior, similar to the Java Application Environment API. An experienced Java developer will rely on a Java API throwing runtime exceptions when bad parameters are supplied. For a Java Card implementation, this will ensure that defective applets can be detected and will facilitate debugging of applets.
- **Size Reduction** Less code is required, resulting in a smaller API implementation compared to one that does explicit checking of parameters.
- **Consistency** A Java Card applet that executes on one Java Card implementation is more likely to exhibit the identical functional behavior on other Java Card implementations.

Please refer to the *Java Card 2.1 API Specification* for more details.

java.lang package changes

Classes `ArithmeticException, ArrayIndexOutOfBoundsException, ArrayStoreException, ClassCastException, IndexOutOfBoundsException, NegativeArraySizeException, NullPointerException`:

Description change: added documentation in the class description explaining that all JCRE owned exception objects are temporary JCRE Entry Point Objects. Also introduced text warning against the storing of references to temporary JCRE Entry Point Objects.

Description change: documentation for all the `java.lang` classes to be similar to the JDK documentation, with the additional statement that these classes form a strict subset of JDK.

Description change: deleted optional behavior to lock up card on exception condition.

Class `SecurityException`:

Description change: documentation for all this class to be similar to the JDK documentation, with the additional statement that this class forms a strict subset of JDK.

Description change: added documentation in the class description explaining that all JCRE owned exception objects are temporary JCRE Entry Point Objects. Also introduced text warning against the storing of references to temporary JCRE Entry Point Objects.

javacard.framework package changes

Class `JCSystem:`

The `System` class has been renamed to `JCSystem`.

Method deleted: `makeTransientObject` has been replaced by transient factory methods.

New method: `makeTransientBooleanArray` is the new `boolean` array transient factory method.

New method: `makeTransientByteArray` is the new `byte` array transient factory method.

New method: `makeTransientShortArray` is the new `short` array transient factory method.

New method: `makeTransientObjectArray` is the new `object` array transient factory method.

New method: `isTransient` now returns transient event type describing when the transient data is reset.

New method: `getAppletShareableInterfaceObject` to request access to shared interface.

New method: `getPreviousContextAID` to determine caller AID.

New method: `lookupAID` to obtain AID of another applet.

Documentation change: `abortTransaction` has been changed to warn against the use of this method when object instantiation is included in the transaction.

New interface: `Shareable` to designate interfaces for sharing with another applet.

Class `AID`:

New method: constructor now public to allow `new` from other package.

Method renamed: `copyTo(byte[], short, byte)` method is renamed as `getBytes(byte[], short, byte)`

Method renamed: `isEqual(byte[], short, byte)` method is renamed as `equals(byte[],short,byte).`

New method: `equals(Object)` to compare AID objects.

New method: `partialEquals(byte[], short, byte)` to compare `AID` bytes against AID objects for partial match.

New method: `RIDEquals(AID otherAID)` to compare RID bytes of `AID` object against RID bytes of another AID object.

Interface PIN:

The abstract `PIN` class has been redefined as an interface.

Interface `ISO7816:`

The constants-only `ISO` class has been redefined as an interface and renamed as `ISO7816`.

New response status code: `SW_APPLET_SELECT_FAILED` to indicate applet selection failure.

Class `OwnerPIN`:

Method renamed: `updateAndUnblock` method has been renamed as `update`.

Documentation Change: clarified documentation in the class description to state the implementation restrictions regarding the creation of transient objects.

Class `ProxyPIN`:

This class has been deleted.

Class `Applet`:

New method: `register( byte[], short, byte )` to allow multiple applet instances.

New method: `getShareableInterfaceObject` to grant access to shared interface.

Change method: `register` method throws `SystemException` if the applet instance AID is in use or already registered.

Change method: `process` method is now `abstract`. An Applet subclass must implement this method.

Documentation Change: added documentation in the `process` method explaining that the APDU object parameter is a temporary JCRE Entry Point Object. Also added text warning against the storing of references to temporary JCRE Entry Point Objects.

Documentation Change: added note in the `process` method to clarify the potential consequences of altering the APDU buffer prior to invoking the `APDU.setIncomingAndReceive` method.

Documentation change: `process` method states that APDU buffer cleared between messages.

Documentation change: `install` method states installation is successful when `register` completes without exception.

Documentation Change: added documentation in the `install` method explaining that the `bArray` byte array parameter is a global array. Also added text warning against the storing of references to global array.

Class `APDU`:

Documentation Change: added documentation in the class description explaining that the APDU object is a temporary JCRE Entry Point Object. Also added text warning against the storing of references to temporary JCRE Entry Point Objects.

Documentation Change: added documentation in the class description as well as the `getBuffer` method to explain that the APDU buffer is a (temporary) global array. Also added text warning against the storing of references to global arrays.

New method: `getProtocol` method to allow legacy applications that modify behavior based on underlying protocol.

Change method: `getInBlockSize` method now returns `short` instead of `byte`.

New method: `setOutgoingNoChaining` method to allow applets to operate with legacy terminals that do not support block chaining.

Documentation change: `setOutgoingLength` method throws `APDUException.BAD_LENGTH` if `len` > 256.

Documentation change: `sendBytes` method throws `APDUException.BUFFER_BOUNDS` not `APDUExceptin.BAD_LENGTH`.

Documentation change: `sendBytes` method throws `APDUException.NO_T0_GETRESPONSE` when T=0 protocol is in use and the CAD does not respond with the expected `GET RESPONSE` command.

Documentation change: `sendBytes` method honors the output mode requested by the `setOutgoingNoChaining` method to not use block chaining for outgoing data transfer.

Documentation change: `waitExtension` throws exception if no chaining output mode requested by the `setOutgoingNoChaining` method.

Documentation change: `receiveBytes` method specifies that the all remaining bytes must be returned if they fit.

Class `APDUException, CardException, CardRuntimeException, ISOException, PINException, SystemException, TransactionException, UserException`:

Documentation Change: added documentation in the class description explaining that all JCRE owned exception objects are temporary JCRE Entry Point Objects. Also added text warning against the storing of references to temporary JCRE Entry Point Objects.

Class `APDUException`:

New exception reason: `NO_T0_GET_RESPONSE` to be thrown by sending methods when the underlying T=0 implementation needs to indicate that the CAD rejected optional data. No further output data or status can be sent. The JCRE needs to re-dispatch the new command received.

Class `SystemException`:

New exception reason: `ILLEGAL_TRANSIENT` to be thrown by transient factory methods to indicate disallowed use.

New exception reason: `NO_RESOURCE` to indicated lack for resources on the card.

Deleted exception reason: `ALREADY_TRANSIENT` reason code. This is no longer applicable.

Class `Util`:

Method changed: `arrayFillNonAtomic` now takes offset, length parameters.

Class `CardException`:

This new class is part of the restructured Exception Class hierarchy. This class defines the `reason` field and accessor methods.

Class `CardRuntimeException`:

This new class is part of the restructured Exception Class hierarchy. This class defines the `reason` field and accessor methods

javacardx.framework package changes

The entire package has been deleted.

javacardx.cryptoEnc package changes

This package has been replaced with the restructured `java.security` and `javacardx.crypto` packages.

javacard.security package (new)

This is a new package containing the following new Interfaces and Classes:

New Interfaces:

`Key SecretKey DESKey PrivateKey PublicKey RSAPrivateKey RSAPrivateCrtKey RSAPublicKey DSAPrivateKey DSAKey`

New Classes:

`KeyBuilder MessageDigest RandomData Signature CryptoException`

javacardx.crypto package

All the Java Card 2.0 classes have been deleted. The following new classes have been added.

Class `Cipher:`

New class to obtain instance of desired cipher implementation

Also base class for cipher implementations.

Interface `KeyEncryption`:

New Interface to allow encrypted key data to be presented to the key set methods.

3. Other Java Card 2.1 Documentation

The *Java Card 2.1 Virtual Machine Specification* describes the subset of the Java virtual machine and language that is supported in the Java Card 2.1 platform.

To further define and explain the Java Card 2.1 API, please refer to *Java Card 2.1 Runtime Environment (JCRE) Specification.* This specification describes the standard runtime environment for the Java Card platform.

The *Java Card 2.1 Virtual Machine Specification* and the *Java Card 2.1 Runtime Environment (JCRE) Specification* are available on the Java Card Technology web page for public access.

Another related document is the *Java Card 2.1 Applet Developer's Guide*, which contains guidelines for a Java Card applet programmer. This document will be available for public access in March, 1999.