Java Card 2.0
Application Programming Interfaces

October 13, 1997
Revision 1.0 Final

Java Card 2.0 API

Table of Contents

Package Index

Java API Packages

- package java.lang

Other Packages

- package javacard.framework
- package javacardx.crypto
- package javacardx.cryptoEnc
- package javacardx.framework

Class Hierarchy

- class java.lang.Object
    - class javacard.framework.AID
    - class javacard.framework.APDU
    - class javacard.framework.Applet
    - class javacardx.framework.File
        - class javacardx.framework.DedicatedFile
            - class javacardx.framework.FileSystem
        - class javacardx.framework.ElementaryFile
            - class javacardx.framework.LinearVariableFile
                - class javacardx.framework.LinearFixedFile
                    - class javacardx.framework.CyclicFile
            - class javacardx.framework.TransparentFile
    - class javacard.framework.ISO
    - class javacardx.crypto.Key
        - class javacardx.crypto.AsymKey
            - class javacardx.crypto.PrivateKey
                - class javacardx.crypto.RSA_CRT_PrivateKey
                - class javacardx.crypto.RSA_PrivateKey
            - class javacardx.crypto.PublicKey
                - class javacardx.crypto.RSA_PublicKey
        - class javacardx.crypto.SymKey
            - class javacardx.crypto.DES3_Key
                - class javacardx.cryptoEnc.DES3_EncKey
            - class javacardx.crypto.DES_Key
                - class javacardx.cryptoEnc.DES_EncKey
    - class javacardx.crypto.MessageDigest
        - class javacardx.crypto.Sha1MessageDigest
    - class javacard.framework.PIN
        - class javacard.framework.OwnerPIN
        - class javacard.framework.ProxyPIN
    - class javacardx.crypto.RandomData
    - class javacard.framework.System
    - class java.lang.Throwable
        - class java.lang.Exception
            - class java.lang.RuntimeException
                - class javacard.framework.APDUException
                - class java.lang.ArithmeticException
                - class java.lang.ArrayStoreException
                - class java.lang.ClassCastException
                - class javacardx.crypto.CryptoException
                - class javacard.framework.ISOException
                - class java.lang.IndexOutOfBoundsException

                - class java.lang.ArrayIndexOutOfBoundsException
            - class java.lang.NegativeArraySizeException
            - class java.lang.NullPointerException
            - class javacard.framework.PINException
            - class java.lang.SecurityException
            - class javacard.framework.SystemException
            - class javacard.framework.TransactionException
        - class javacard.framework.UserException
    - class javacard.framework.Util

package java.lang

Class Index

- Object
- Throwable

Exception Index

- ArithmeticException
- ArrayIndexOutOfBoundsException
- ArrayStoreException
- ClassCastException
- Exception
- IndexOutOfBoundsException
- NegativeArraySizeException
- NullPointerException
- RuntimeException
- SecurityException

Class java.lang.ArithmeticException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----java.lang.ArithmeticException
```

public class **ArithmeticException**
extends RuntimeException

`ArithmeticException` is thrown on an illegal arithmetic operation. The JCRE may choose to mute the card instead.

Constructor Index

o **ArithmeticException**(short)
Constructs an ArithmeticException with the specified reason.

Constructors

o **ArithmeticException**

```
public ArithmeticException(short reason)
```

Constructs an ArithmeticException with the specified reason.

Parameters:
reason - the reason for the exception.

Class java.lang.ArrayIndexOutOfBoundsException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----java.lang.IndexOutOfBoundsException
                                   |
                                   +----java.lang.ArrayIndexOutOfBoundsException
```

public class **ArrayIndexOutOfBoundsException**
extends IndexOutOfBoundsException

`ArrayIndexOutOfBoundsException` is thrown on an attempt to access an element within an array with an index not within the bounds of the array. The JCRE may choose to mute the card instead.

Constructor Index

o **ArrayIndexOutOfBoundsException**(short)
Constructs an ArrayIndexOutOfBoundsException with the specified reason.

Constructors

o **ArrayIndexOutOfBoundsException**

```
public ArrayIndexOutOfBoundsException(short reason)
```

Constructs an ArrayIndexOutOfBoundsException with the specified reason.

Parameters:
reason - the reason for the exception.

Class java.lang.ArrayStoreException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----java.lang.ArrayStoreException
```

public class **ArrayStoreException**
extends RuntimeException

`ArrayStoreException` is thrown to indicate that an attempt has been made to store the wrong type of object into an array of objects. The JCRE may choose to mute the card instead.

Constructor Index

o **ArrayStoreException**(short)
Constructs an ArrayStoreException with the specified reason.

Constructors

o **ArrayStoreException**

```
 public ArrayStoreException(short reason)
```

Constructs an ArrayStoreException with the specified reason.

Parameters:
reason - the reason for the exception.

Class java.lang.ClassCastException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----java.lang.ClassCastException
```

public class **ClassCastException**
extends RuntimeException

`ClassCastException` is thrown on an attempt to cast an instance of a class to another class that is not allowed.The JCRE may choose to mute the card instead.

Constructor Index

o **ClassCastException**(short)
Constructs a ClassCastException with the specified reason.

Constructors

o **ClassCastException**

```
 public ClassCastException(short reason)
```

Constructs a ClassCastException with the specified reason.

Parameters:
reason - the reason for the exception.

Class java.lang.Exception

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
```

public class **Exception**
extends Throwable

`Exception` represents a general Java Card exception. This is the base class for all checked exceptions in Java Card.

Constructor Index

o **Exception**()
  Constructs an Exception instance with reason = 0.
o **Exception**(short)
  Constructs an Exception instance with the specified reason.

Method Index

o **throwIt**(short)
  Throws the re-usable JCRE instance of Exception with the specified reason.

Constructors

o **Exception**

```
public Exception()
```

Constructs an Exception instance with reason = 0. To conserve on resources use `throwIt()` to re-use the JCRE instance of this class.

o **Exception**

```
public Exception(short reason)
```

Constructs an Exception instance with the specified reason. To conserve on resources use `throwIt()` to re-use the JCRE instance of this class.

Parameters:
reason - the reason for the exception.

Methods

o **throwIt**

```
public static void throwIt(short reason) throws Exception
```

Throws the re-usable JCRE instance of Exception with the specified reason. Subclasses must override this method to throw the subclass instance instead. Additionally, the overriding method must change the `throws` clause in the method declaration to specify the subclass.

Parameters:
reason - the reason for the exception.
Throws: Exception
always.

Class java.lang.IndexOutOfBoundsException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----java.lang.IndexOutOfBoundsException
```

public class **IndexOutOfBoundsException**
extends RuntimeException

`IndexOutOfBoundsException` is thrown to indicate that an index of some sort (such as to an array) is out of range. The JCRE may choose to mute the card instead.

Constructor Index

o **IndexOutOfBoundsException**(short)
Constructs an IndexOutOfBoundsException with the specified reason.

Constructors

o **IndexOutOfBoundsException**

```
public IndexOutOfBoundsException(short reason)
```

Constructs an IndexOutOfBoundsException with the specified reason.

Parameters:
reason - the reason for the exception.

Class java.lang.NegativeArraySizeException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----java.lang.NegativeArraySizeException
```

public class **NegativeArraySizeException**
extends RuntimeException

`NegativeArraySizeException` is thrown on an attempt to create an array with a negative size. The JCRE may choose to mute the card instead.

Constructor Index

o **NegativeArraySizeException**(short)
Constructs a NegativeArraySizeException with the specified reason.

Constructors

o **NegativeArraySizeException**

```
public NegativeArraySizeException(short reason)
```

Constructs a NegativeArraySizeException with the specified reason.

Parameters:
reason - the reason for the exception.

Class java.lang.NullPointerException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----java.lang.NullPointerException
```

public class **NullPointerException**
extends RuntimeException

`NullPointerException` is thrown on an attempt to dereference a null object reference.The JCRE may choose to mute the card instead.

Constructor Index

o **NullPointerException**(short)
Constructs a NullPointerException with the specified reason.

Constructors

o **NullPointerException**

```
 public NullPointerException(short reason)
```

Constructs a NullPointerException with the specified reason.

Parameters:
reason - the reason for the exception.

Class java.lang.Object

```
java.lang.Object
```

public class **Object**

Class Object is the root of the Java Card class hierarchy. Every class has Object as a superclass. All objects, including arrays, implement the methods of this class.

Constructor Index

o **Object**()

Method Index

o **equals**(Object)
Compares two Objects for equality.

Constructors

o **Object**

```
 public Object()
```

Methods

o **equals**

```
 public boolean equals(Object obj)
```

Compares two Objects for equality.

Parameters:
obj - the reference object with which to compare.
Returns:
true if this object is the same as the obj argument; false otherwise.

All Packages Class Hierarchy This Package Previous Next Index

Class java.lang.RuntimeException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
```

public class **RuntimeException**
extends Exception

`RuntimeException` represents a general Runtime exception in Java Card.

Constructor Index

o **RuntimeException**()
  Constructs a Runtime exception instance with reason = 0.
o **RuntimeException**(short)
  Constructs a Runtime exception instance with the specified reason.

Method Index

o **throwIt**(short)
  Throws the JCRE instance of the Runtime exception with the specified reason.

Constructors

o **RuntimeException**

```
public RuntimeException()
```

Constructs a Runtime exception instance with reason = 0. To conserve on resources use `throwIt()` to re-use the JCRE instance of this class.

o **RuntimeException**

```
public RuntimeException(short reason)
```

Constructs a Runtime exception instance with the specified reason. To conserve on resources use `throwIt()` to re-use the JCRE instance of this class.

Parameters:
reason - the reason for the exception.

Methods

o **throwIt**

```
public static void throwIt(short reason)
```

Throws the JCRE instance of the Runtime exception with the specified reason.

Parameters:
reason - the reason for the exception.
Throws: RuntimeException
always.

All Packages Class Hierarchy This Package Previous Next Index

Class java.lang.SecurityException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----java.lang.SecurityException
```

public class **SecurityException**
extends RuntimeException

`SecurityException` represents an object access violation. This exception is thrown when an attempt is made to illegally access an object belonging to a another applet. The JCRE may choose to mute the card instead.

Constructor Index

o **SecurityException**(short)
Constructs a SecurityException with the specified reason.

Constructors

o **SecurityException**

```
public SecurityException(short reason)
```

Constructs a SecurityException with the specified reason.

Parameters:
reason - the reason for the exception.

Class java.lang.Throwable

```
java.lang.Object
   |
   +----java.lang.Throwable
```

public class **Throwable**
extends Object

The Throwable class is the superclass of all errors and exceptions in the Java Card subset. Only objects that are instances of this class (or of one of its subclasses) are thrown by the JCRE or can be thrown by the Java throw statement. Similarly, only this class or one of its subclasses can be the argument type in a catch clause.

Variable Index

o **reason**
The reason for the exception.

Constructor Index

o **Throwable**()

Method Index

o **getReason**()
Returns the reason for the exception.
o **setReason**(short)
Sets the reason for the exception.

Variables

o **reason**

```
protected short reason
```

The reason for the exception.

Constructors

o **Throwable**

```
public Throwable()
```

Methods

o **getReason**

```
public short getReason()
```

Returns the reason for the exception.

Returns:
the reason for the exception.

o **setReason**

```
public void setReason(short reason)
```

Sets the reason for the exception.

Parameters:
reason - the exception reason.

All Packages Class Hierarchy This Package Previous Next Index

All Packages Class Hierarchy Index

package javacard.framework

Class Index

- AID
- APDU
- Applet
- ISO
- OwnerPIN
- PIN
- ProxyPIN
- System
- Util

Exception Index

- APDUException
- ISOException
- PINException
- SystemException
- TransactionException
- UserException

All Packages Class Hierarchy This Package Previous Next Index

Class javacard.framework.AID

```
java.lang.Object
   |
   +----javacard.framework.AID
```

public final class **AID**
extends Object

This class encapsulates the Application Identifier(AID) associated with an applet. It contains a byte array of 5..16 bytes as defined in ISO 7816-5.

The JCRE creates instances of AID class using the package private constructor to uniquely identify and manage every applet loaded on the card. The JCRE shares these unique instances with all applets on the card.

Applets can use the AID object to uniquely identify another applet on the card. An applet can obtain a reference its unique AID object by using `System.getAID()`. To compare two AID objects, it is sufficient to compare references to them.

Method Index

o **copyTo**(byte[], short)
Called to obtain a copy of the byte array within AID object.
o **isEqual**(byte[], short, byte)
Checks if the specified AID byte array is the same as `this` object's byte array.

Methods

o **copyTo**

```
public byte copyTo(byte dest[],
                   short offset)
```

Called to obtain a copy of the byte array within AID object.

Parameters:
dest - byte array to copy to.
offset - within dest to start the copy.
Returns:
the length of the AID byte array.

o **isEqual**

```
public boolean isEqual(byte bArray[],
                       short offset,
                       byte length)
```

Checks if the specified AID byte array is the same as `this` object's byte array.

Parameters:
bArray - to compare against
offset - within bArray to begin
length - of AID byte array
Returns:
true if equal, false otherwise.

All Packages Class Hierarchy This Package Previous Next Index

All Packages Class Hierarchy This Package Previous Next Index

Class javacard.framework.APDU

```
java.lang.Object
   |
   +----javacard.framework.APDU
```

public final class **APDU**
extends Object

Application Protocol Data Unit (APDU) is the communication format between the card and the off-card applications. The format of the APDU is defined in ISO specification 7816-4.

This class only supports messages which conform to the structure of command and response defined in ISO 7816-4. The behavior of messages which use proprietary structure of messages (e.g with header CLA byte in range D0-FE) is undefined. Additionally, this class does not support extended length fields.

APDU objects are owned by the JCRE. The APDU class maintains a byte array buffer which is used to transfer incoming APDU header and data bytes as well as outgoing data. The buffer length must be at least 37 bytes.

The applet receives an APDU instance to process from the JCRE in the `Applet.process(APDU)` method, and the first five bytes [CLA, INS, P1, P2, P3] are available in the APDU buffer.

The APDU class API is designed to be transport protocol independent. In other words, applets can use the same APDU methods regardless of whether the underlying protocol in use is T=0 or T=1 (as defined in ISO 7816-3).

Depending on the size of the incoming APDU data, it may not fit inside the buffer and may need to be read in portions by the applet. Depending on the size of the outgoing response APDU data, it may not fit inside the buffer and may need to be written in portions by the applet. The APDU class has methods to facilitate this.

For sending large byte arrays as response data, the APDU class provides a special method `sendBytesLong()` which manages the APDU buffer.

```
 // The purpose of this example is to show most of the methods
 // in use and not to depict any particular APDU processing
public void process(APDU apdu){
  // ...
  byte[] buffer = apdu.getBuffer();
  byte cla = buffer[ISO.OFFSET_CLA];
  byte ins = buffer[ISO.OFFSET_INS];
  ...
  // assume this command has incoming data
  // Lc tells us the incoming apdu command length
  short bytesLeft = (short) (buffer[ISO.OFFSET_LC] & 0x00FF);
  if (bytesLeft < ...) ISOException.throwIt( ISO.SW_WRONG_LENGTH );
  short readCount = apdu.setIncomingAndReceive();
  while ( bytesLeft > 0){
      // process buffer[5..readCount+4];
      bytesLeft -= readCount;
      readCount = apdu.receiveBytes ( ISO.OFFSET_CDATA );
```

```
      }
  //
  //...
  //
  // Note that for a short response as in the case illustrated here
  // the three APDU method calls shown : setOutgoing(),setOutgoingLength() & sendBytes()
  // could be replaced by one APDU method call : setOutgoingAndSend().
  // construct the reply APDU
  short le = apdu.setOutgoing();
  if (le < 2) ISOException.throwIt( ISO.SW_WRONG_LENGTH );
  apdu.setOutgoingLength( (short)3 );
  // build response data in apdu.buffer[ 0.. outCount-1 ];
  buffer[0] = (byte)1; buffer[1] = (byte)2; buffer[3] = (byte)3;
  apdu.sendBytes ( (short)0 , (short)3 );
  // return good complete status 90 00
  }
```

Method Index

o **getBuffer**()
    Returns the APDU buffer byte array.
o **getInBlockSize**()
    Returns the configured incoming block size.
o **getNAD**()
    Returns the T=1 transport protocol Node Address byte, NAD.T=0 returns 0.
o **receiveBytes**(short)
    Gets as many data bytes as will safely fit (without buffer overflow) in the APDU buffer at the specified offset `bOff`.
o **sendBytes**(short, short)
    Sends `len` more bytes from `apdu.buffer` at specified offset `bOff`.
o **sendBytesLong**(byte[], short, short)
    Sends `len` more bytes from `outData` at specified offset `bOff`.
o **setIncomingAndReceive**()
    This is the primary receive method.
o **setOutgoing**()
    This method is used to set the data transfer direction to outbound and to obtain the expected length of response (Le).
o **setOutgoingAndSend**(short, short)
    This is the "convenience" send method.
o **setOutgoingLength**(short)
    Sets the expected length of response data.
o **wait**()
    Requests additional processsing time from Terminal.

Methods

o **getBuffer**

```
 public byte[] getBuffer()
```

Returns the APDU buffer byte array.

Returns:
byte array containing the APDU buffer

o **getInBlockSize**

```
public static byte getInBlockSize()
```

Returns the configured incoming block size. In T=1, this corresponds to the maximum size of incoming data blocks from the terminal, IFSC (information field size for ICC). T=0, returns 1. IFSC is defined in ISO 7816-3. This information may be used to ensure that there is enough space remaining in the APDU buffer when `receiveBytes()` is invoked.

Notes:
- *On* `receiveBytes()` *the bOff param should account for this potential blocksize.*
- *T=0 will return 1.*

Returns:
incoming block size setting.

o **getNAD**

```
public byte getNAD()
```

Returns the T=1 transport protocol Node Address byte, NAD.T=0 returns 0. This may be used as additional information to maintain multiple contexts.

Note:
- *T=0 will return 0.*

Returns:
NAD transport byte as defined in ISO 7816-3.

o **setOutgoing**

```
public short setOutgoing() throws APDUException
```

This method is used to set the data transfer direction to outbound and to obtain the expected length of response (Le).

Notes.
- *The remaining incoming data if any, will be discarded.*
- *T=0 (Case 4) will return 256.*
- *The APDU buffer at offset 0 will be used to read the unread incoming data.*

Returns:
the Le.
Throws: APDUException
with the following reason codes:
- APDUException.ILLEGAL_USE if method already invoked.
- APDUException.IO_ERROR on I/O error.

o **setOutgoingLength**

```
public void setOutgoingLength(short len) throws APDUException
```

Sets the expected length of response data. Default is 0.

Notes:
- *Used in T=0 (Case 4) protocol to prompt terminal for GET RESPONSE command (processed by APDU).*
- *In T=0 (Case 2), if expected length different, prompts for correct length GET RESPONSE (processed by APDU).*

Parameters:
len - the length of response data.
Throws: APDUException
with the following reason codes:
- APDUException.ILLEGAL_USE if `setOutgoing()` not called or this method already invoked.
- APDUException.IO_ERROR on I/O error.

o **receiveBytes**

```
public short receiveBytes(short bOff) throws APDUException
```

Gets as many data bytes as will safely fit (without buffer overflow) in the APDU buffer at the specified offset `bOff`.

Notes:
- *The space in the buffer must allow for incoming block size (see* `getInBlockSize()` *).*
- *In T=1, the terminal may send in less than InBlockSize bytes.*
- *User must manage the APDU buffer.*

Parameters:
bOff - the offset into APDU buffer.
Returns:
number of bytes read. 0 if no bytes available.
Throws: APDUException
with the following reason codes:
- APDUException.ILLEGAL_USE if `setIncomingAndReceive()` not called.
- APDUException.BUFFER_BOUNDS if not enough buffer space for incoming block size.
- APDUException.IO_ERROR on I/O error.

o **setIncomingAndReceive**

```
public short setIncomingAndReceive() throws APDUException
```

This is the primary receive method. Indicates that this APDU has incoming data. This method gets as many bytes as will safely fit (without buffer overflow) in the APDU buffer following the header.

Notes:
- *Used in T=0 (Case 3 or 4) protocol to assume P3 param is Lc.*
- *Data is read into the buffer at offset 5.*
- *In T=1, the terminal may send in less than InBlockSize bytes.*

- *This method sets the transfer direction to be inbound and calls* `receiveBytes(5)`.
- *This method may only be called once.*

Returns:
number of bytes read. returns 0 if no bytes available.
Throws: APDUException
with the following reason codes:
- APDUException.ILLEGAL_USE if `setIncomingAndReceive()` already invoked.
- APDUException.IO_ERROR on I/O error.

o **sendBytes**

```
public void sendBytes(short bOff,
                      short len) throws APDUException
```

Sends `len` more bytes from `apdu.buffer` at specified offset `bOff`.

User must manage the APDU buffer.

If the last of the response is being sent, the APDU buffer must not be altered upon return from this method. This allows the implementation to reduce protocol overhead by transmitting the last part of the response alongwith the status bytes.

Parameters:
bOff - the offset into APDU buffer.
len - the length of the data in bytes to send.
Throws: APDUException
with the following reason codes:
- APDUException.ILLEGAL_USE if `setOutgoing()` not called or `setOutgoingAndSend()` previously invoked or response byte count exeeded.
- APDUException.BAD_LENGTH if `bOff` or `len` is too large.
- APDUException.IO_ERROR on I/O error.

o **sendBytesLong**

```
public void sendBytesLong(byte outData[],
                          short bOff,
                          short len) throws APDUException
```

Sends `len` more bytes from `outData` at specified offset `bOff`.

If the last of the response is being sent, the APDU buffer must not be altered upon return from this call. This allows the implementation to reduce protocol overhead by transmitting the last part of the response alongwith the status bytes.

JCRE will manage the APDU buffer.

Notes:
- *Note that the actual data transmission may take place on return from Applet.*

Parameters:
outData - the large byte array source.
bOff - the offset into OutData array.

len - the bytelength of the data to send.
Throws: APDUException
with the following reason codes:
- APDUException.ILLEGAL_USE if response byte count exeeded.
- APDUException.IO_ERROR on I/O error.

o **setOutgoingAndSend**

```
public void setOutgoingAndSend(short bOff,
                               short len) throws APDUException, ISOException
```

This is the "convenience" send method. It provides for the most efficient way to send a short response which fits in the buffer and needs the least protocol overhead. This method is a combination of `setOutgoing(), setOutgoingLength( len )` followed by `sendBytes ( bOff, len )`. In addition, once this method is invoked, `sendBytes` and `sendBytesLong` methods cannot be invoked and the APDU buffer must not be altered.

Sends `len` byte response from `apdu.buffer` at specified offset `bOff`.

Notes:
- *If the expected response length, Le is less than* `len`, `ISOException(ISO.SW_CORRECT_LENGTH_00+len)` *is thrown.*
- *No other APDU send methods can be invoked.*
- *The APDU buffer must not be altered.*
- *The actual data transmission may only take place on return from Applet.*

Parameters:
bOff - the offset into APDU buffer.
len - the bytelength of the data to send.
Throws: APDUException
with the following reason codes:
- APDUException.ILLEGAL_USE if `setOutgoing()` or `setOutgoingAndSend()` previously invoked or response byte count exeeded.
- APDUException.IO_ERROR on I/O error.

Throws: ISOException
with the following reason codes:
- (ISO.SW_CORRECTED_LENGTH_00+len) if Terminal expected length (Le)is less than sending length, len.

o **wait**

```
public void wait()
```

Requests additional processsing time from Terminal. The implementation should ensure that this method needs to be invoked only under unusual conditions requiring excessive processing times.

Notes:
- *In T=0,a NULL procedure byte is sent to reset the work waiting time (see ISO 7816-3).*
- *In T=1, the implementation needs to request the same T=0 work waiting time quantum by sending a T=1 request for wait time extension(see ISO 7816-3).*
- *If the implementation uses an automatic timer mechanism instead, this method may be a NOP.*

Class javacard.framework.APDUException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----javacard.framework.APDUException
```

public class **APDUException**
extends RuntimeException

`APDUException` represents an APDU related exception.

Table APDUException

reason	**Description**
ILLEGAL_USE	APDU Illegal Use
BUFFER_BOUNDS	APDU buffer bounds error
BAD_LENGTH	APDU outGoingLength inconsistency
IO_ERROR	APDU I/O Error

Variable Index

o **BAD_LENGTH**
o **BUFFER_BOUNDS**
o **ILLEGAL_USE**
o **IO_ERROR**

Constructor Index

o **APDUException**(short)
Constructs an APDUException.

Method Index

o **throwIt**(short)
Throws the JCRE instance of APDUException with the specified reason.

Variables

o **ILLEGAL_USE**

```
public static final short ILLEGAL_USE
```

o **BUFFER_BOUNDS**

```
public static final short BUFFER_BOUNDS
```

o **BAD_LENGTH**

```
public static final short BAD_LENGTH
```

o **IO_ERROR**

```
public static final short IO_ERROR
```

Constructors

o **APDUException**

```
public APDUException(short reason)
```

Constructs an APDUException. To conserve on resources use `throwIt()` to re-use the JCRE instance of this class.

Parameters:
reason - the reason for the exception.

Methods

o **throwIt**

```
public static void throwIt(short reason)
```

Throws the JCRE instance of APDUException with the specified reason.

Parameters:
reason - the reason for the exception.
Throws: APDUException
always.

Class javacard.framework.Applet

```
java.lang.Object
   |
   +----javacard.framework.Applet
```

public abstract class **Applet**
extends Object

This abstract class defines an applet in a smart card.

The `Applet` class should be extended by any applet wishing to be loaded onto, installed into and executed on a Java Card compliant smart card.

Example usage of `Applet`

```
public class MyApplet extends javacard.framework.Applet{
  static byte b[];
  private static final byte MIN_APDU_BUFLEN = (byte) 32;
  public static void install( APDU apdu ) throws ISOException {
    // make all my allocations here, so I do not run
    // out of memory later
    MyApplet me = new MyApplet();
    b = new byte[100];
    // check length of APDU buffer
    if ( apdu.getBuffer().length >= MIN_APDU_BUFLEN ) me.register();
    else ISOException.throwIt(ISO.SW_FUNC_NOT_SUPPORTED);
  }
  public boolean select(){
    // selection initialization
    b[17] = 42;
    return true;
  }
  public void process(APDU apdu) throws ISOException{
  byte[] buffer = apdu.getBuffer();
  // .. process the incoming data and reply
  if ( buffer[ISO.OFFSET_CLA] == (byte)00 ) {
      switch ( buffer[ISO.OFFSET_INS] ) {
          case ISO.INS_SELECT:
              ...
              // send response data to select command
              short Le =  apdu.setOutgoing();
              // assume data containing response bytes in replyData[] array.
              if ( Le < ..) ISOException.throwIt( ISO.SW_WRONG_LENGTH);
              apdu.setOutgoingLength( (short)replyData.length );
              apdu.sendBytesLong(replyData, (short) 0, (short)replyData.length);
              break;
          case ...
          }
      }
  }
}
```

Constructor Index

o **Applet**()

Method Index

o **deselect**()
    Called by the JCRE to inform this currently selected applet that another (or the same) applet will be selected.
o **install**(APDU)
    Installs this applet.
o **process**(APDU)
    Processes an incoming APDU.
o **register**()
    Register an applet with the JCRE.
o **select**()
    Called by the JCRE to inform this applet that it has been selected.

Constructors

o **Applet**

```
protected Applet()
```

Methods

o **install**

```
public static void install(APDU apdu) throws ISOException
```

Installs this applet. Any specific installation calls by the applet should be issued here, e.g., calls to check JCRE resources, such as:

```
private static final byte MIN_APDU_BUFLEN = (byte) 32;
...
if ( apdu.getBuffer().length >= MIN_APDU_BUFLEN ) ..
else ... // error
```

This method is called by the JCRE at install time. Upon normal return from this method the JCRE sends ISO 7816-4 defined good complete status (90 00) in APDU response. If this method throws an ISOExcepton the JCRE sends the associated reason code as the response status instead.

The five header bytes of the APDU are available in `APDU.buffer[0..4]` at the time this method is called.

The implementation of this method provided by `Applet` class throws an ISOException(ISO.SW_FUNC_NOT_SUPPORTED).

Notes:
- *Normal return signals to the JCRE that this applet should be installed.*
- *APDU buffer[5..] is undefined and should not be read or written at this time.*

Parameters:
    apdu - the incoming APDU containing the INSTALL command.
Throws: ISOException
    with the response bytes per ISO 7816-4
See Also:
    APDU

o **process**

```
public void process(APDU apdu) throws ISOException
```

Processes an incoming APDU. An Applet is expected to perform the action requested and return response data if any to the terminal.

Upon normal return from this method the JCRE sends ISO 7816-4 defined good complete status (90 00) in APDU response. If this method throws an ISOExcepton the JCRE sends the associated reason code as the response status instead.

The five header bytes of the APDU are available in `APDU.buffer[0..4]` at the time this method is called.

Notes:
- *APDU buffer[5..] is undefined and should not be read or written at this time.*

Parameters:
    apdu - the incoming APDU
Throws: ISOException
    with the response bytes per ISO 7816-4
See Also:
    APDU

o **select**

```
public boolean select()
```

Called by the JCRE to inform this applet that it has been selected.

It is called when a SELECT command is received and the applet is selected. A subclass of `Applet` should override this method if it wants to perform any initialization that may be required to process APDU messages that may follow. This method returns a boolean to indicate that it is ready to accept incoming APDUs via it's `process` method. If this method returns false, it indicates to the JCRE that this Applet declines to be selected.

The implementation of this method provided by `Applet` class returns true.

Returns:
    true to indicate success, false otherwise.

o **deselect**

```
public void deselect()
```

Called by the JCRE to inform this currently selected applet that another (or the same) applet will be selected. It is called when a SELECT command is received by the JCRE. This method is invoked prior to some `select` method being invoked.

A subclass of `Applet` should override this method if it has any cleanup or bookkeeping work to be performed before another applet is selected.

The implementation of this method provided by `Applet` class does nothing.

Note:

- *Unchecked exceptions thrown by this method are ignored.*
- *This method is NOT called on reset or power loss.*

o **register**

```
 protected final void register()
```

Register an applet with the JCRE. This method should be called during `install` to register this Applet subclass instance with the JCRE.

Class javacard.framework.ISO

```
java.lang.Object
   |
   +----javacard.framework.ISO
```

---

public class **ISO**
extends Object

`ISO` encapsulates constants related to ISO 7816-3 and ISO 7816-4. `ISO` class contains only static fields.

The static fields with `SW_` prefixes define constants for the ISO 7816-4 defined response status word. The fields which use the `_00` suffix require the low order byte to be customized appropriately e.g (ISO.CORRECT_LENGTH_00 + 0x0025).

The static fields with `OFFSET_` prefixes define constants to be used to index into the APDU buffer byte array to access ISO 7816-4 defined header information.

Variable Index

o **OFFSET_CDATA**
  APDU command data offset : CDATA = 5
o **OFFSET_CLA**
  APDU header offset : CLA = 0
o **OFFSET_INS**
  APDU header offset : INS = 1
o **OFFSET_LC**
  APDU header offset : LC = 4
o **OFFSET_P1**
  APDU header offset : P1 = 2
o **OFFSET_P2**
  APDU header offset : P2 = 3
o **SW_BYTES_REMAINING_00**
  Response status : Response bytes remaining = 0x6100
o **SW_CLA_NOT_SUPPORTED**
  Response status : CLA value not supported = 0x6E00
o **SW_CONDITIONS_NOT_SATISFIED**
  Response status : Conditions of use not satisfied = 0x6985
o **SW_CORRECT_LENGTH_00**
  Response status : Correct Expected Length (Le) = 0x6C00
o **SW_DATA_INVALID**
  Response status : Data invalid = 0x6984
o **SW_FILE_FULL**
  Response status : Not enough memory space in the file = 0x6A84

o **SW_FILE_INVALID**
Response status : File invalid = 0x6983
o **SW_FILE_NOT_FOUND**
Response status : File not found = 0x6A82
o **SW_FUNC_NOT_SUPPORTED**
Response status : Function not supported = 0x6A81
o **SW_INCORRECT_P1P2**
Response status : Incorrect parameters (P1,P2) = 0x6A86
o **SW_INS_NOT_SUPPORTED**
Response status : INS value not supported = 0x6D00
o **SW_NO_ERROR**
Response status : No Error = (short)0x9000
o **SW_PIN_REQUIRED**
Response status : PIN required = 0x6982
o **SW_RECORD_NOT_FOUND**
Response status : Record not found = 0x6A83
o **SW_SECURITY_STATUS_NOT_SATISFIED**
Response status : Security condition not satisfied = 0x6982
o **SW_UNKNOWN**
Response status : No precise diagnosis = 0x6F00
o **SW_WRONG_DATA**
Response status : Wrong data = 0x6A80
o **SW_WRONG_LENGTH**
Response status : Wrong length = 0x6700
o **SW_WRONG_P1P2**
Response status : Incorrect parameters (P1,P2) = 0x6B00

Variables

o **SW_NO_ERROR**

```
public static final short SW_NO_ERROR
```

Response status : No Error = (short)0x9000

o **SW_BYTES_REMAINING_00**

```
public static final short SW_BYTES_REMAINING_00
```

Response status : Response bytes remaining = 0x6100

o **SW_WRONG_LENGTH**

```
public static final short SW_WRONG_LENGTH
```

Response status : Wrong length = 0x6700

o **SW_PIN_REQUIRED**

```
public static final short SW_PIN_REQUIRED
```

Response status : PIN required = 0x6982

o **SW_FILE_INVALID**

```
public static final short SW_FILE_INVALID
```

Response status : File invalid = 0x6983

o **SW_DATA_INVALID**

```
public static final short SW_DATA_INVALID
```

Response status : Data invalid = 0x6984

o **SW_CONDITIONS_NOT_SATISFIED**

```
public static final short SW_CONDITIONS_NOT_SATISFIED
```

Response status : Conditions of use not satisfied = 0x6985

o **SW_WRONG_DATA**

```
public static final short SW_WRONG_DATA
```

Response status : Wrong data = 0x6A80

o **SW_FUNC_NOT_SUPPORTED**

```
public static final short SW_FUNC_NOT_SUPPORTED
```

Response status : Function not supported = 0x6A81

o **SW_FILE_NOT_FOUND**

```
public static final short SW_FILE_NOT_FOUND
```

Response status : File not found = 0x6A82

o **SW_RECORD_NOT_FOUND**

```
public static final short SW_RECORD_NOT_FOUND
```

Response status : Record not found = 0x6A83

o **SW_INCORRECT_P1P2**

```
public static final short SW_INCORRECT_P1P2
```

Response status : Incorrect parameters (P1,P2) = 0x6A86

o **SW_WRONG_P1P2**

```
public static final short SW_WRONG_P1P2
```

Response status : Incorrect parameters (P1,P2) = 0x6B00

o **SW_CORRECT_LENGTH_00**

```
public static final short SW_CORRECT_LENGTH_00
```

Response status : Correct Expected Length (Le) = 0x6C00

o **SW_INS_NOT_SUPPORTED**

```
public static final short SW_INS_NOT_SUPPORTED
```

Response status : INS value not supported = 0x6D00

o **SW_CLA_NOT_SUPPORTED**

```
public static final short SW_CLA_NOT_SUPPORTED
```

Response status : CLA value not supported = 0x6E00

o **SW_UNKNOWN**

```
public static final short SW_UNKNOWN
```

Response status : No precise diagnosis = 0x6F00

o **SW_FILE_FULL**

```
public static final short SW_FILE_FULL
```

Response status : Not enough memory space in the file = 0x6A84

o **SW_SECURITY_STATUS_NOT_SATISFIED**

```
public static final short SW_SECURITY_STATUS_NOT_SATISFIED
```

Response status : Security condition not satisfied = 0x6982

o **OFFSET_CLA**

```
public static final byte OFFSET_CLA
```

APDU header offset : CLA = 0

o **OFFSET_INS**

```
public static final byte OFFSET_INS
```

APDU header offset : INS = 1

o **OFFSET_P1**

```
public static final byte OFFSET_P1
```

APDU header offset : P1 = 2

o **OFFSET_P2**

```
public static final byte OFFSET_P2
```

APDU header offset : P2 = 3

o **OFFSET_LC**

```
public static final byte OFFSET_LC
```

APDU header offset : LC = 4

o **OFFSET_CDATA**

```
public static final byte OFFSET_CDATA
```

APDU command data offset : CDATA = 5

All Packages Class Hierarchy This Package Previous Next Index

Class javacard.framework.ISOException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----javacard.framework.ISOException
```

public class **ISOException**
extends RuntimeException

`ISOException` class encapsulates an ISO 7816-4 response status word as its `reason` code.

Constructor Index

o **ISOException**(short)
Constructs an ISOException instance with the specified status word.

Method Index

o **throwIt**(short)
Throws the JCRE instance of the ISOexception class with the specified status word.

Constructors

o **ISOException**

```
public ISOException(short sw)
```

Constructs an ISOException instance with the specified status word. To conserve on resources use `throwIt()` to re-use the JCRE instance of this class.

Parameters:
sw - the ISO 7816-4 defined status word

Methods

o **throwIt**

```
public static void throwIt(short sw)
```

Throws the JCRE instance of the ISOexception class with the specified status word.

Parameters:
sw - ISO 7816-4 defined status word
Throws: ISOException
always.

Class javacard.framework.OwnerPIN

```
java.lang.Object
   |
   +----javacard.framework.PIN
           |
           +----javacard.framework.OwnerPIN
```

public class **OwnerPIN**
extends PIN

This class represents an Owner PIN. It derives from abstract PIN class. It provides the ability to update the PIN and thus owner functionality.

The implementaion of this class must protect against attacks based on program flow prediction.

The protected methods `getValidatedFlag` and `setValidatedFlag` allow a subclass of this class to optimize the storage for the validated boolean state.

Instances of this class are only suitable for sharing when there exists a trust relationship amongst the applets. A typical shared usage would use an OwnerPIN class instance and a shared ProxyPIN class instance.

Constructor Index

o **OwnerPIN**(byte, byte)
  Constructor.

Method Index

o **check**(byte[], short, byte)
  Compares `pin` against the PIN value.
o **getTriesRemaining**()
  Returns the number of times remaining that an incorrect PIN can be presented before the PIN is blocked.
o **getValidatedFlag**()
  This protected method returns the validated flag.
o **isValidated**()
  Returns true if a valid PIN has been presented since the last card reset or last call to `reset()`.
o **reset**()
  If the validated flag is set, this method resets it.
o **resetAndUnblock**()
  This method resets the validated flag and resets the PIN try counter to the value of the PIN try limit.
o **setValidatedFlag**(boolean)
  This protected method sets the value of the validated flag.
o **updateAndUnblock**(byte[], short, byte)
  This method sets a new value for the PIN and resets the PIN try counter to the value of the PIN try limit.

Constructors

o **OwnerPIN**

```
public OwnerPIN(byte tryLimit,
                byte maxPINSize) throws PINException
```

Constructor. Allocates a new PIN instance.

Parameters:
  tryLimit - the maximum number of times an incorrect PIN can be presented.
  maxPINSize - the maximum allowed PIN size.
Throws: PINException
  with the following reason codes:
  - PINException.ILLEGAL_VALUE on illegal parameter.

Methods

o **getValidatedFlag**

```
protected boolean getValidatedFlag()
```

This protected method returns the validated flag.

Returns:
  the boolean state of the PIN validated flag.

o **setValidatedFlag**

```
protected void setValidatedFlag(boolean value)
```

This protected method sets the value of the validated flag.

Parameters:
  value - the new value for the validated flag.

o **getTriesRemaining**

```
public byte getTriesRemaining()
```

Returns the number of times remaining that an incorrect PIN can be presented before the PIN is blocked.

Returns:
  the number of times remaining
Overrides:
  getTriesRemaining in class PIN

o **check**

```
public boolean check(byte pin[],
                     short offset,
                     byte length)
```

Compares `pin` against the PIN value. If they match and the PIN is not blocked, it sets the validated flag and resets the try counter to its maximum. If it does not match, it decrements the try counter, and if the counter has reached zero, blocks the PIN.

Parameters:
pin - the PIN value being checked
offset - the starting offset in the pin array
length - the length of pin.
Returns:
true if the PIN matches; false otherwise
Overrides:
check in class PIN

o **isValidated**

```
public boolean isValidated()
```

Returns true if a valid PIN has been presented since the last card reset or last call to `reset()`.

Returns:
true if validated; false otherwise
Overrides:
isValidated in class PIN

o **reset**

```
public void reset()
```

If the validated flag is set, this method resets it. If the validated flag is not set, this method does nothing.

Overrides:
reset in class PIN

o **updateAndUnblock**

```
public void updateAndUnblock(byte pin[],
                             short offset,
                             byte length) throws PINException
```

This method sets a new value for the PIN and resets the PIN try counter to the value of the PIN try limit. It also resets the validated flag.

Parameters:
pin - the bytearray containing the new pin value
offset - the starting offset in the pin array
length - the length of the new pin.
Throws: PINException
with the following reason codes:

- PINException.ILLEGAL_VALUE on illegal parameter.

o **resetAndUnblock**

```
public void resetAndUnblock()
```

This method resets the validated flag and resets the PIN try counter to the value of the PIN try limit. This method is used by the owner to re-enable the blocked PIN.

All Packages Class Hierarchy This Package Previous Next Index

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacard.framework.PIN

```
java.lang.Object
   |
   +----javacard.framework.PIN
```

public abstract class **PIN**
extends Object

This class represents a PIN. It maintains these internal values:

- PIN value
- try limit, the maximum number of times an incorrect PIN can be presented before the PIN is blocked. When the PIN is blocked, it cannot be validated even on valid PIN presentation.
- max PIN size, the maximum length of PIN allowed
- try counter, the remaining number of times an incorrect PIN presentation is permitted
- validated flag, true if a valid PIN has been presented. This flag is reset on every card reset.

This class does not make any assumptions about where the data for the PIN comparison is stored.

An owner subclass of this abstract class must provide a way to initialize/update the PIN value.The implementaion of the subclass must protect against attacks based on program flow prediction.

A typical card global PIN usage will combine an instance of OwnerPIN class and a shared instance of the ProxyPIN class. The OwnerPIN instance would be manipulated only by the owner who has update privilege. All others would access the global PIN functionality via the ProxyPIN instance.

Constructor Index

o **PIN**()
  Constructs a PIN instance.

Method Index

o **check**(byte[], short, byte)
  Compares `pin` against the PIN value.
o **getTriesRemaining**()
  Returns the number of times remaining that an incorrect PIN can be presented before the PIN is blocked.
o **isValidated**()
  Returns true if a valid PIN has been presented since the last card reset or last call to `reset()`.
o **reset**()
  If the validated flag is set, this method resets it.

Constructors

o **PIN**

```
public PIN()
```

Constructs a PIN instance.

Methods

o **getTriesRemaining**

```
public abstract byte getTriesRemaining()
```

Returns the number of times remaining that an incorrect PIN can be presented before the PIN is blocked.

Returns:
the number of times remaining

o **check**

```
public abstract boolean check(byte pin[],
                              short offset,
                              byte length)
```

Compares `pin` against the PIN value. If they match and the PIN is not blocked, it sets the validated flag and resets the try counter to its maximum. If it does not match, it decrements the try counter, and if the counter has reached zero, blocks the PIN.

Parameters:
pin - the PIN value being checked
offset - the starting offset in the pin array
length - the length of pin.
Returns:
true if the PIN matches; false otherwise

o **isValidated**

```
public abstract boolean isValidated()
```

Returns true if a valid PIN has been presented since the last card reset or last call to `reset()`.

Returns:
true if validated; false otherwise

o **reset**

```
public abstract void reset()
```

If the validated flag is set, this method resets it. If the validated flag is not set, this method does nothing.

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacard.framework.PINException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----javacard.framework.PINException
```

public class **PINException**
extends RuntimeException

`PINException` represents a PIN access-related exception. This class also provides a resource-saving mechanism for user exceptions by re-using a JCRE instance.

Table PINException

reason	**Description**
ILLEGAL_VALUE	Illegal parameter value

Variable Index

o **ILLEGAL_VALUE**

Constructor Index

o **PINException**(short)
Constructs a PINException.

Method Index

o **throwIt**(short)
Throws the JCRE instance of PINException with the specified reason.

Variables

o **ILLEGAL_VALUE**

```
public static final short ILLEGAL_VALUE
```

Constructors

o **PINException**

```
public PINException(short reason)
```

Constructs a PINException. To conserve on resources use `throwIt()` to re-use the JCRE instance of this class.

Parameters:
reason - the reason for the exception.

Methods

o **throwIt**

```
public static void throwIt(short reason)
```

Throws the JCRE instance of PINException with the specified reason.

Parameters:
reason - the reason for the exception.
Throws: PINException
always.

Class javacard.framework.ProxyPIN

```
java.lang.Object
   |
   +----javacard.framework.PIN
           |
           +----javacard.framework.ProxyPIN
```

---

public class **ProxyPIN**
extends PIN

This class represents a proxy for some real PIN instance. It maintains a reference to that PIN instance. All methods of ProxyPIN refer the operation to the real PIN.

Constructor Index

o **ProxyPIN**(PIN)
Constructor.

Method Index

o **check**(byte[], short, byte)
Compares `pin` against the PIN value.
o **getTriesRemaining**()
Returns the number of times remaining that an incorrect PIN can be presented before the PIN is blocked.
o **isValidated**()
Returns true if a valid PIN has been presented since the last card reset or last successful call to `reset()`.
o **reset**()
If the validated flag is set, this method resets it.

Constructors

o **ProxyPIN**

```
public ProxyPIN(PIN realPIN) throws PINException
```

Constructor. Allocates a ProxyPIN object to the real PIN instance.

Parameters:
PIN - the real PIN instance.
Throws: PINException
with the following reason codes:

- PINException.ILLEGAL_VALUE on illegal parameter.

Methods

o **getTriesRemaining**

```
public final byte getTriesRemaining()
```

Returns the number of times remaining that an incorrect PIN can be presented before the PIN is blocked.

Returns:
the number of times remaining
Overrides:
getTriesRemaining in class PIN

o **check**

```
public final boolean check(byte pin[],
                           short offset,
                           byte length)
```

Compares `pin` against the PIN value. If they match and the PIN is not blocked, it sets the validated flag and resets the try counter to its maximum. If it does not match, it decrements the try counter, and if the counter has reached zero, blocks the PIN.

Parameters:
pin - the PIN value being checked
offset - the starting offset in the pin array
length - the length of pin.
Returns:
true if the PIN matches; false otherwise
Overrides:
check in class PIN

o **isValidated**

```
public final boolean isValidated()
```

Returns true if a valid PIN has been presented since the last card reset or last successful call to `reset()`.

Returns:
true if validated; false otherwise
Overrides:
isValidated in class PIN

o **reset**

```
public final void reset()
```

If the validated flag is set, this method resets it. If the validated flag is not set, this method does nothing.

Overrides:
reset in class PIN

All Packages Class Hierarchy This Package Previous Next Index

Class javacard.framework.System

```
java.lang.Object
   |
   +----javacard.framework.System
```

public final class **System**
extends Object

The `System` class is a centralized point of applet execution, resource management and security in the smart card. All methods in `System` class are static methods.

The System class is used to control the persistence and transience of objects. Objects are persistent by default. The term *persistent* does not mean there is an object-oriented database on the card or that objects are serialized/deserialized. It means that objects and their values persist from one CAD session to the next, indefinitely. Persistent object values are updated atomically using transactions.

Objects can be made *transient* with the `makeTransient` method. The values of transient objects do not persist, and are reset to a default state at specified intervals. Updates to the values of transient objects are not atomic and are not affected by transactions.

On startup, JCRE initializes the commit buffer (used for `beginTransaction()` ...).

Variable Index

o **TRANSIENT_APDU**
Transience duration attribute is applet ADPU process.
o **TRANSIENT_NONE**
Transience duration attribute is NONE.
o **TRANSIENT_SELECTION**
Transience duration attribute is applet selection.
o **TRANSIENT_SESSION**
Transience duration attribute is CAD session.

Method Index

o **abortTransaction**()
Aborts the atomic transaction.
o **beginTransaction**()
Begins an atomic transaction.
o **commitTransaction**()
Commits an atomic transaction.
o **getAID**()
Returns the unique Applet Identifier (AID) object associated with the current applet execution context.

o **getMaxCommitCapacity**()
Returns the total number of bytes in the commit buffer.
o **getTransactionDepth**()
Returns the current transaction nesting depth level.
o **getUnusedCommitCapacity**()
Returns the number of bytes left in the commit buffer.
o **getVersion**()
Returns the current major and minor version of the Java Card API.
o **isTransient**(Object)
Used to check if the object is transient and determine its transience duration attribute.
o **makeTransient**(Object, byte)
Called to make the specified object transient with the specified transience duration attribute.
o **share**(Object)
Makes the specified object instance available for access from any installed applet on the card.
o **share**(Object, AID)
Makes the specified object instance available for access from the applet identified by the specified AID object.

Variables

o **TRANSIENT_SESSION**

```
public static byte TRANSIENT_SESSION
```

Transience duration attribute is CAD session. The contents of the object are reset at the end of each CAD session, or when the card is removed from the CAD.

o **TRANSIENT_SELECTION**

```
public static byte TRANSIENT_SELECTION
```

Transience duration attribute is applet selection. The contents of the object are reset when the object's owning applet is deselected.

o **TRANSIENT_APDU**

```
public static byte TRANSIENT_APDU
```

Transience duration attribute is applet ADPU process. The contents of the object are reset when the method Applet.process() returns.

o **TRANSIENT_NONE**

```
public static byte TRANSIENT_NONE
```

Transience duration attribute is NONE. The object is not transient.

Methods

o **share**

```
public static void share(Object object,
                         AID otherAID) throws SystemException, SecurityException
```

Makes the specified object instance available for access from the applet identified by the specified AID object. Only the owner of the object instance can call this method.

Parameters:
object - the object which we want to share.
otherAID - identifies the other applet to share with.
Throws: SecurityException
if the object is not owned by the current execution context.
Throws: SystemException
with the following reason codes:

- SystemException.ILLEGAL_VALUE if otherAID parameter is invalid.

o **share**

```
public static void share(Object object) throws SecurityException
```

Makes the specified object instance available for access from any installed applet on the card. Only the owner of the object instance can call this method.

Parameters:
object - the object which we want to share with all others.
Throws: SecurityException
if the object is not owned by the current execution context.

o **isTransient**

```
public static byte isTransient(Object object)
```

Used to check if the object is transient and determine its transience duration attribute.

Parameters:
object - the object being queried.
Returns:
transience duration attribute. The possible values are listed in `makeTransient()`.
See Also:
makeTransient

o **makeTransient**

```
public static void makeTransient(Object object,
                                 byte duration)
```

Called to make the specified object transient with the specified transience duration attribute. This method throws a SystemException if the specified object already has a transient attribute not equal to TRANSIENT_NONE.

Note:

- *The total storage space for transient objects may be limited. If sufficient space is not available to store the transient object a SystemException(NO_TRANSIENT_SPACE) may be thrown during object access.*
- *To reduce volatile memory requirements try using shorter transience durations.*

Parameters:
object - the object to be made available in volatile memory.
duration - transient duration attribute to assign the object.

Table Transient duration attribute.

duration	Description
System.TRANSIENT_SESSION	the transience duration is a CAD session.
System.TRANSIENT_SELECTION	the transience duration is applet selection.
System.TRANSIENT_APDU	the transience duration is applet APDU process.
System.TRANSIENT_NONE	the object is not transient.

Throws: SystemException
with the following reason codes:

- SystemException.ALREADY_TRANSIENT if the specified object does not have a TRANSIENT_NONE attribute.
- SystemException.ILLEGAL_VALUE if the duration parameter is invalid.

o **getVersion**

```
public static short getVersion()
```

Returns the current major and minor version of the Java Card API.

Returns:
version number as byte.byte (major.minor)

o **getAID**

```
public static AID getAID()
```

Returns the unique Applet Identifier (AID) object associated with the current applet execution context. When a virtual method is invoked on an object, the applet execution context is changed to correspond to the applet which owns that object; when that method returns, the previous context is restored.

Invocations of static methods have no effect on the applet execution context. The applet execution context and sharing status of an object together determine if access to an object is permissible.

Returns:
the AID object reference.

o **beginTransaction**

```
public static void beginTransaction() throws TransactionException
```

Begins an atomic transaction. The JCRE maintains a commit buffer into which data is written so that JCRE always can guarantee, at commit time, that everything in the buffer is written, or nothing at all. If a transaction is already in progress (transactionDepth != 0), a TransactionException is thrown.

Throws: TransactionException
with the following reason codes:
- TransactionException.IN_PROGRESS if a transaction is already in progress.

See Also:
commitTransaction, abortTransaction

o **abortTransaction**

```
public static void abortTransaction() throws TransactionException
```

Aborts the atomic transaction. The contents of the commit buffer is discarded.

Throws: TransactionException
with the following reason codes:
- TransactionException.NOT_IN_PROGRESS if a transaction is not in progress.

See Also:
beginTransaction, commitTransaction

o **commitTransaction**

```
public static void commitTransaction() throws TransactionException
```

Commits an atomic transaction. The contents of commit buffer is atomically commited. If a transaction is not in progress (transactionDepth == 0) then a TransactionException is thrown.

Throws: TransactionException
with the following reason codes:
- TransactionException.NOT_IN_PROGRESS if a transaction is not in progress.

See Also:
beginTransaction, abortTransaction

o **getTransactionDepth**

```
public static byte getTransactionDepth()
```

Returns the current transaction nesting depth level. At present, only 1 transaction can be in progress at a time.

Returns:
1 if transaction in progress, 0 if not.

o **getUnusedCommitCapacity**

```
public static short getUnusedCommitCapacity()
```

Returns the number of bytes left in the commit buffer.

Returns:
the number of bytes left in the commit buffer

See Also:
getMaxCommitCapacity

o **getMaxCommitCapacity**

```
public static short getMaxCommitCapacity()
```

Returns the total number of bytes in the commit buffer. This is approximately the maximum number of bytes of persistent data which can be modified during a transaction. However, the transaction subsystem requires additional bytes of overhead data to be included in the commit buffer, and this depends on the number of fields modified and the implementation of the transaction subsystem. The application cannot determine the actual maximum amount of data which can be modified during a transaction without taking these overhead bytes into consideration.

Returns:
the total number of bytes in the commit buffer

See Also:
getUnusedCommitCapacity

All Packages Class Hierarchy This Package Previous Next Index

All Packages Class Hierarchy This Package Previous Next Index

Class javacard.framework.SystemException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----javacard.framework.SystemException
```

public class **SystemException**
extends RuntimeException

`SystemException` represents a System class related exception.

Table SystemException

reason	**Description**
ILLEGAL_VALUE	Illegal parameter value
ALREADY_TRANSIENT	Object is already transient
NO_TRANSIENT_SPACE	No room in volatile memory for object

Variable Index

o **ALREADY_TRANSIENT**
o **ILLEGAL_VALUE**
o **NO_TRANSIENT_SPACE**

Constructor Index

o **SystemException**(short)
Constructs a SystemException.

Method Index

o **throwIt**(short)
Throws the JCRE instance of SystemException with the specified reason.

Variables

o **ILLEGAL_VALUE**

```
public static final short ILLEGAL_VALUE
```

o **ALREADY_TRANSIENT**

```
public static final short ALREADY_TRANSIENT
```

o **NO_TRANSIENT_SPACE**

```
public static final short NO_TRANSIENT_SPACE
```

Constructors

o **SystemException**

```
public SystemException(short reason)
```

Constructs a SystemException. To conserve on resources use `throwIt()` to re-use the JCRE instance of this class.

Parameters:
reason - the reason for the exception.

Methods

o **throwIt**

```
public static void throwIt(short reason)
```

Throws the JCRE instance of SystemException with the specified reason.

Parameters:
reason - the reason for the exception.
Throws: SystemException
always.

All Packages Class Hierarchy This Package Previous Next Index

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacard.framework.TransactionException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----javacard.framework.TransactionException
```

public class **TransactionException**
extends RuntimeException

`TransactionException` represents an exception in the transaction subsystem.

Table TransactionException

reason	**Description**
IN_PROGRESS	beginTransaction called when already in progress
NOT_IN_PROGRESS	commit/abortTransaction called when not in progress
BUFFER_FULL	commit buffer is full
INTERNAL_FAILURE	internal JCRE problem (fatal error)

Variable Index

o **BUFFER_FULL**
o **IN_PROGRESS**
o **INTERNAL_FAILURE**
o **NOT_IN_PROGRESS**

Constructor Index

o **TransactionException**(short)
Constructs a TransactionException with the specified reason.

Method Index

o **throwIt**(short)
Throws the JCRE instance of TransactionException with the specified reason.

Variables

o **IN_PROGRESS**

`public static final short IN_PROGRESS`

o **NOT_IN_PROGRESS**

`public static final short NOT_IN_PROGRESS`

o **BUFFER_FULL**

`public static final short BUFFER_FULL`

o **INTERNAL_FAILURE**

`public static final short INTERNAL_FAILURE`

Constructors

o **TransactionException**

`public TransactionException(short reason)`

Constructs a TransactionException with the specified reason. To conserve on resources use `throwIt()` to re-use the JCRE instance of this class.

Methods

o **throwIt**

`public static void throwIt(short reason)`

Throws the JCRE instance of TransactionException with the specified reason.

Throws: TransactionException
always.

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacard.framework.UserException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----javacard.framework.UserException
```

public class **UserException**
extends Exception

`UserException` represents a User exception. This class also provides a resource-saving mechanism for user exceptions by re-using a JCRE instance.

Constructor Index

o **UserException**()
  Constructs a UserException with reason = 0.
o **UserException**(short)
  Constructs a UserException with the specified reason.

Method Index

o **throwIt**(short)
  Throws the re-usable JCRE instance of UserException with the specified reason.

Constructors

o **UserException**

```
public UserException()
```

Constructs a UserException with reason = 0. To conserve on resources use `throwIt()` to re-use the JCRE instance of this class.

o **UserException**

```
public UserException(short reason)
```

Constructs a UserException with the specified reason. To conserve on resources use `throwIt()` to re-use the JCRE instance of this class.

Parameters:
  reason - the reason for the exception.

Methods

o **throwIt**

```
public static void throwIt(short reason) throws UserException
```

Throws the re-usable JCRE instance of UserException with the specified reason.

Parameters:
  reason - the reason for the exception.
Throws: UserException
  always.

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacard.framework.Util

```
java.lang.Object
   |
   +----javacard.framework.Util
```

public class **Util**
extends Object

The `util` class contains common utility functions. Some of the methods may be implemented as native functions for performance reasons. All methods in `Util` class are static methods.

Some methods of `Util` namely `arrayCopy`, `arrayCopyNonAtomic`, `arrayFillNonAtomic` and `setShort` refer to the persistence of array objects. The term *persistent* does not mean that there is an object-oriented database on the card or that arrays are serialized/deserialized. It means that arrays and their values persist from one CAD session to the next, indefinitely.

The System class is used to control the persistence and transience of objects.

See Also:
System

Method Index

o **arrayCompare**(byte[], short, byte[], short, short)
Compares an array from the specified source array, beginning at the specified position, with the specified position of the destination array from left to right.
o **arrayCopy**(byte[], short, byte[], short, short)
Copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array.
o **arrayCopyNonAtomic**(byte[], short, byte[], short, short)
Copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array (non-atomically).
o **arrayFillNonAtomic**(byte[], byte)
Fills the byte array (non-atomically) with the specified value.
o **getShort**(byte[], short)
Concatenates two bytes in a byte array to form a short value
o **makeShort**(byte, byte)
Concatenates the two parameter bytes to form a short value
o **setShort**(byte[], short, short)
Deposits the short value as two successive bytes at the specified offset in the byte array.

Methods

o **arrayCopy**

```
public static final void arrayCopy(byte src[],
                                   short srcOff,
                                   byte dest[],
                                   short destOff,
                                   short length)
```

Copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array.

Note:

- *If the src and dest arguments refer to the same array object, then the copying is performed as if the components at positions srcOff through srcOff+length-1 were first copied to a temporary array with length components and then the contents of the temporary array were copied into positions destOff through destOff+length-1 of the argument array.*
- *If the destination array is persistent, the entire copy is performed atomically.*
- *The copy operation is subject to atomic commit capacity limitations.*

Parameters:
src - source byte array.
srcOff - offset within source byte array to start copy from.
dest - destination byte array.
destOff - offset within destination byte array to start copy into.
length - byte length to be copied.
See Also:
getUnusedCommitCapacity

o **arrayCopyNonAtomic**

```
public static final void arrayCopyNonAtomic(byte src[],
                                            short srcOff,
                                            byte dest[],
                                            short destOff,
                                            short length)
```

Copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array (non-atomically).

This method does not use the transaction facility during the copy operation.

Thus, this method is suitable for use only when the contents of the destination array can be left in a partially modified state in the event of a power loss in the middle of the copy operation.

Note:

- *If the src and dest arguments refer to the same array object, then the copying is performed as if the components at positions srcOff through srcOff+length-1 were first copied to a temporary array with length components and then the contents of the temporary array were copied into positions destOff through destOff+length-1 of the argument array.*
- *If power is lost during the copy operation and the destination array is persistent, a partially changed destination array could result.*
- *The copy length parameter is not constrained by the atomic commit capacity limitations.*

Parameters:
src - source byte array.
srcOff - offset within source byte array to start copy from.
dest - destination byte array.
destOff - offset within destination byte array to start copy into.
length - byte length to be copied.
See Also:
getUnusedCommitCapacity

o **arrayFillNonAtomic**

```
public static final void arrayFillNonAtomic(byte bArray[],
                                            byte bValue)
```

Fills the byte array (non-atomically) with the specified value.

This method does not use the transaction facility during the fill operation.

Thus, this method is suitable for use only when the contents of the byte array can be left in a partially filled state in the event of a power loss in the middle of the fill operation.

Note:
- *If power is lost during the copy operation and the byte array is persistent, a partially changed byte array could result.*
- *The length parameter is not constrained by the atomic commit capacity limitations.*

Parameters:
bArray - the byte array.
bValue - the value to fill the byte array with.
See Also:
getUnusedCommitCapacity

o **arrayCompare**

```
public static byte arrayCompare(byte src[],
                                short srcOff,
                                byte dest[],
                                short destOff,
                                short length)
```

Compares an array from the specified source array, beginning at the specified position, with the specified position of the destination array from left to right. Returns the ternary result of the comparison : less than(-1), equal(0) or greater than(1).

Parameters:
src - source byte array.
srcOff - offset within source byte array to start compare.
dest - destination byte array.
destOff - offset within destination byte array to start compare.
length - byte length to be compared.
Returns:
the result of the comparison as follows:
- 0 if identical
- -1 if the first miscomparing byte in source array is less than that in destination array,

- 1 if the first miscomparing byte in source array is greater that that in destination array.

o **makeShort**

```
public static final short makeShort(byte b1,
                                    byte b2)
```

Concatenates the two parameter bytes to form a short value

Parameters:
b1 - the first byte (high order byte).
b2 - the second byte (low order byte).
Returns:
theShort - the concatenated result

o **getShort**

```
public static final short getShort(byte bArray[],
                                   short bOff)
```

Concatenates two bytes in a byte array to form a short value

Parameters:
bArray - byte array.
bOff - offset within byte array containing first byte (the high order byte).
Returns:
theShort - the concatenated result

o **setShort**

```
public static final void setShort(byte bArray[],
                                  short bOff,
                                  short sValue)
```

Deposits the short value as two successive bytes at the specified offset in the byte array.

Parameters:
bArray - byte array.
bOff - offset within byte array to deposit the first byte (the high order byte).
sValue - the short value to set into array.

Note:
- *If the byte array is persistent, this operation is performed atomically.*

See Also:
getUnusedCommitCapacity

All Packages Class Hierarchy Index

package javacardx.framework

Class Index

- CyclicFile
- DedicatedFile
- ElementaryFile
- File
- FileSystem
- LinearFixedFile
- LinearVariableFile
- TransparentFile

All Packages Class Hierarchy This Package Previous Next Index

Class javacardx.framework.CyclicFile

```
java.lang.Object
   |
   +----javacardx.framework.File
           |
           +----javacardx.framework.ElementaryFile
                   |
                   +----javacardx.framework.LinearVariableFile
                           |
                           +----javacardx.framework.LinearFixedFile
                                   |
                                   +----javacardx.framework.CyclicFile
```

public class **CyclicFile**
extends LinearFixedFile

Cyclic fixed-length record file. Records are organized as a ring (cyclic structure), with fixed and equal record size. The number of records in a CyclicFile is defined at file creation time and can not be changed.

Records are numbered in the reverse order as they were inserted into the file. Thus the record inserted last is record number one.

See Also:
LinearFixedFile, LinearVariableFile

Constructor Index

o **CyclicFile**(short, byte, byte)
Constructor.

Method Index

o **addRecord**(byte[])
Not allowed for cyclic files.
o **addRecord**(short)
Not allowed for cyclic files.
o **findRecord**(byte, byte, byte, byte)
Find the record.
o **getNewFirstRecord**()
Get the next unused record or recycle the oldest record as the new most recent record (record number 1).
o **getRecord**(byte)
Get the record byte array for the specified record.
o **increaseMaxNumRecords**(byte)
Not allowed for cyclic files.

Constructors

o **CyclicFile**

```
public CyclicFile(short FID,
                  byte maxNumRecords,
                  byte recordLength)
```

Constructor.

Parameters:
FID - the file’s 16-bit FID
maxNumRecords - the maximum number of records in this file
recordLength - the fixed record length for this file

Methods

o **getRecord**

```
public byte[] getRecord(byte recordNum)
```

Get the record byte array for the specified record. Records are numbered in the reverse order that they were updated in the file. Record number is in the range from 1 to the number of records in the file.

Parameters:
recordNum - the record number. The most recently updated record is record number one.
Returns:
record (or null)
Overrides:
getRecord in class LinearVariableFile

o **findRecord**

```
public byte findRecord(byte direction,
                       byte currentRecNumber,
                       byte firstByte,
                       byte secondByte)
```

Find the record. Using the specified direction and current record number as the starting point, find the record for which first and second byte match `firstByte` and `secondByte` specified in the parameter. Records are numbered in the reverse order that they were updated in the file. (See Annex C of ISO 7816-4 for details)

Parameters:
direction - one of the DIRECTION_xxx constants. see LinearVariableFile
firstByte - if non-0, the record’s first byte must match this value; if 0, any value of the record’s first byte matches.
secondByte - if non-0, the record’s second byte must match this value; if 0, any value of the record’s second byte matches.
currentRecNumber - current record number. If 0, the current record is undefined.
Returns:
the record number, or 0 if the record is not found

Overrides:
findRecord in class LinearVariableFile
See Also:
LinearVariableFile

o **getNewFirstRecord**

```
public byte[] getNewFirstRecord()
```

Get the next unused record or recycle the oldest record as the new most recent record (record number 1).

Returns:
record, a reference to the next unused record or the oldest record in the file. Its contents must be updated by the caller.

o **increaseMaxNumRecords**

```
public boolean increaseMaxNumRecords(byte number) throws ISOException
```

Not allowed for cyclic files.

Throws: ISOException
always throws ISOException.
- ISOException.reason = ISO.SW_FUNC_NOT_SUPPORTED

Overrides:
increaseMaxNumRecords in class LinearVariableFile

o **addRecord**

```
public void addRecord(byte record[]) throws ISOException
```

Not allowed for cyclic files.

Throws: ISOException
always throws ISOException.
- ISOException.reason = ISO.SW_FUNC_NOT_SUPPORTED

Overrides:
addRecord in class LinearFixedFile

o **addRecord**

```
public void addRecord(short length) throws ISOException
```

Not allowed for cyclic files.

Throws: ISOException
always throws ISOException.
- ISOException.reason = ISO.SW_FUNC_NOT_SUPPORTED

Overrides:
addRecord in class LinearFixedFile

Class javacardx.framework.DedicatedFile

```
java.lang.Object
   |
   +----javacardx.framework.File
           |
           +----javacardx.framework.DedicatedFile
```

public class **DedicatedFile**
extends File

Dedicated file. A DedicatedFile contains zero or more other file objects (DFs and/or EFs).

Variable Index

o **FIND_ANY**
  Selection mode parameter used with the findFile method.
o **FIND_CHILD**
  Selection mode parameter used with the findFile method.
o **FIND_CHILD_DF**
  Selection mode parameter used with the findFile method.
o **FIND_CHILD_EF**
  Selection mode parameter used with the findFile method.

Constructor Index

o **DedicatedFile**(short, byte[], byte)

Method Index

o **addChildFile**(File)
  Add (append) a new child file to this DedicatedFile.
o **findDedicatedFile**(byte[], short, byte)
  Under this DF, find the DF with the specified name.
o **findElementaryFile**(byte)
  Under this DF, find the EF with the specified SFI.
o **findFile**(byte, short)
  According to the findType, find the file with the specified FID.
o **getChildFile**(byte)
  Get the File object for the specified child file.
o **getMaxChildFiles**()
  Get the maximum number of child files in this DF.
o **getName**()
  Get the file’s name

o **getNumChildFiles**()
    Get the actual number of child files in this DF.
o **increaseMaxChildFiles**(byte)
    Increase the maximum number of child files in this DF.

Variables

o **FIND_ANY**

```
public static final byte FIND_ANY
```

Selection mode parameter used with the findFile method. See findFile for details

o **FIND_CHILD_DF**

```
public static final byte FIND_CHILD_DF
```

Selection mode parameter used with the findFile method. See findFile for details

o **FIND_CHILD_EF**

```
public static final byte FIND_CHILD_EF
```

Selection mode parameter used with the findFile method. See findFile for details

o **FIND_CHILD**

```
public static final byte FIND_CHILD
```

Selection mode parameter used with the findFile method. See findFile for details

Constructors

o **DedicatedFile**

```
public DedicatedFile(short FID,
                     byte name[],
                     byte maxChildFiles)
```

Parameters:
    FID - the file’s 16-bit FID
    name - the name byte array of this file (or null if none)
    maxChildFiles - the maximum number of child files for this DF

Methods

o **getName**

```
public byte[] getName()
```

Get the file’s name

Returns:
    name or null if name is absent.

o **getMaxChildFiles**

```
public byte getMaxChildFiles()
```

Get the maximum number of child files in this DF.

Returns:
    maxChildFiles

o **increaseMaxChildFiles**

```
public boolean increaseMaxChildFiles(byte number)
```

Increase the maximum number of child files in this DF.

Parameters:
    number - increase the maximum number of child files to this number
Returns:
    true if the increase was successful, false otherwise

o **getNumChildFiles**

```
public byte getNumChildFiles()
```

Get the actual number of child files in this DF.

Returns:
    numChildFiles

o **getChildFile**

```
public File getChildFile(byte childNum)
```

Get the File object for the specified child file. Child files are numbered in the order that they were added to the file.

Parameters:
    childNum - the index (first child = 1) of the child file.
Returns:
    the File object (or null)

o **findDedicatedFile**

```
public DedicatedFile findDedicatedFile(byte data[],
                                       short offset,
                                       byte length)
```

Under this DF, find the DF with the specified name.

Parameters:
    data - a byte array containing the name
    offset - byte offset of name in data
    length - length of name in data
Returns:
    the DF selected or null if the DF is not found

o **findElementaryFile**

```
public ElementaryFile findElementaryFile(byte SFI)
```

Under this DF, find the EF with the specified SFI.

Parameters:
SFI - the short file identifier
Returns:
the EF selected or null

o **findFile**

```
public File findFile(byte findType,
                     short FID) throws ISOException
```

According to the findType, find the file with the specified FID. The FIND_xxx constants allow different ways to find a file.

FIND_xxx
- FIND_ANY: Among this DF's parent, siblings and direct children, find a File whose FID matches the given FID
- FIND_CHILD_EF: find an ElementaryFile under this DF whose FID matches the given FID
- FIND_CHILD_DF: find a DedicatedFile under this DF whose FID matches the given FID
- FIND_CHILD: find a child file under this DF whose FID matches the given FID

Parameters:
findType - one of the FIND_xxx constants
FID - the file identifier
Returns:
the File found or null

o **addChildFile**

```
public void addChildFile(File child) throws ISOException
```

Add (append) a new child file to this DedicatedFile.

Parameters:
child - the reference to the child file.
Throws: ISOException
if action fails.
- ISOException.reason = FileSystem.SW_FILE_FULL, if the maximum number of child files for this DF is exceeded.
- ISOException.reason = ISO.SW_CONDITIONS_NOT_SATISFIED, if a condition is not satisfied for adding a new file under this DF. For example, if the new child file's FID is not unique under this DF or if the new child file is a DedicatedFile and its DFname is not unique under this DF.

Class javacardx.framework.ElementaryFile

```
java.lang.Object
   |
   +----javacardx.framework.File
           |
           +----javacardx.framework.ElementaryFile
```

public abstract class **ElementaryFile**
extends File

This is the abstract base class for all elementary files (EFs). For simplicity, the SFI of an EF is the last 5 bits of the FID.

Method Index

o **getSFI**()
Get this file's 5-bit SFI.

Methods

o **getSFI**

```
public byte getSFI()
```

Get this file's 5-bit SFI. The SFI is the last 5 bits of the FID.

Returns:
SFI

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacardx.framework.File

```
java.lang.Object
   |
   +----javacardx.framework.File
```

public abstract class **File**
extends Object

This is the abstract base class for all files (DFs and EFs) in an applet's file system. See `FileSystem` class and ISO 7816-4 for additional details.

All files have:

- a FID (16-bit file identifer)
- a parent DF (which is null if the file has no parent)
- external read/write security attributes

Since an explicit security model is not defined in 7816-4, this class defines a simple yet extensible scheme. Each file has two attributes, one for "external read access" (such as a READ RECORD command) and one for "external write access" (such as a WRITE BINARY command). In each attribute the user can set one of the ALLOW_xxx values to specify what conditions must be true in order to allow that type of access (see tables below).

Table - Access Attributes

Constant	**Description**
ACCESS_READ	External read
ACCESS_WRITE	External write

Table - Allow Types

Constant	**Description**
ALLOW_ANY	Any external access allowed
ALLOW_AUTH1	External access allowed only if Auth1 flag is true
ALLOW_AUTH2	External access allowed only if Auth2 flag is true
ALLOW_NONE	No external access allowed

For example, ALLOW_ANY for the read attribute means that this file can be read externally at any time. ALLOW_NONE for the write attribute means that this file can never be written externally.

The two Auth flags are defined in the `FileSystem` class and allow for a certain amount of applet customization. When a security attribute is set to ALLOW_AUTH1 or ALLOW_AUTH2, the access is allowed only if the appropriate Auth flags maintained by the FileSystem is true. For example, an applet may set Auth1 when a valid PIN is presented. After that point, all files with ALLOW_AUTH1 in the read attribute can now be read externally.

Note that this security checking is done programatically and is not enforced by the VM. That is, the `FileSystem.readRecord` method will perform read access checking on the accessed file. But internal applet access to a EF or DF is not checked unless the applet specifically does so, using the `isAllowed` method in this class.

Variable Index

o **ACCESS_READ**
  read access attribute
o **ACCESS_WRITE**
  write access attribute
o **ALLOW_ANY**
  allow any access
o **ALLOW_AUTH1**
  allow access if AUTH1 flag in FileSystem is true
o **ALLOW_AUTH2**
  allow access if AUTH2 flag in FileSystem is true
o **ALLOW_NONE**
  allow no external access

Method Index

o **getFCI**()
  Get this file's FCI (if any).
o **getFID**()
  Get this file's 16-bit FID.
o **getFileSystem**()
  Get the file system object (if any) which this file belongs to
o **getParent**()
  Get this file's parent DF if any.
o **getSecurity**(byte)
  Get this file's external read or write security.
o **isAllowed**(byte)
  Check this file's external read or write security.
o **setFCI**(byte[])
  Set this file's FCI.
o **setSecurity**(byte, byte)
  Set this file's external read or write security.

Variables

o **ACCESS_READ**

```
public static final byte ACCESS_READ
```

read access attribute

o **ACCESS_WRITE**

```
public static final byte ACCESS_WRITE
```

write access attribute

o **ALLOW_ANY**

```
public static final byte ALLOW_ANY
```

allow any access

o **ALLOW_AUTH1**

```
public static final byte ALLOW_AUTH1
```

allow access if AUTH1 flag in FileSystem is true

o **ALLOW_AUTH2**

```
public static final byte ALLOW_AUTH2
```

allow access if AUTH2 flag in FileSystem is true

o **ALLOW_NONE**

```
public static final byte ALLOW_NONE
```

allow no external access

Methods

o **getFID**

```
public short getFID()
```

Get this file’s 16-bit FID.

Returns:
FID

o **getParent**

```
public DedicatedFile getParent()
```

Get this file’s parent DF if any.

Returns:
parent DF (or null)

o **getFCI**

```
public byte[] getFCI()
```

Get this file’s FCI (if any).

Returns:
the FCI byte array (or null)

o **setFCI**

```
public void setFCI(byte FCI[])
```

Set this file’s FCI.

Parameters:
FCI - the byte array containing the FCI

o **getSecurity**

```
public byte getSecurity(byte access)
```

Get this file’s external read or write security.

Parameters:
access - ACCESS_READ or ACCESS_WRITE
Returns:
one of the ALLOW_xxx constants

o **setSecurity**

```
public void setSecurity(byte access,
                        byte allow)
```

Set this file’s external read or write security.

Parameters:
access - ACCESS_READ or ACCESS_WRITE
allow - one of the ALLOW_xxx constants

o **getFileSystem**

```
public FileSystem getFileSystem()
```

Get the file system object (if any) which this file belongs to

Returns:
the filesystem object or null if this file is not attached to a file system

o **isAllowed**

```
public boolean isAllowed(byte access)
```

Check this file's external read or write security. This method always returns true for ALLOW_ANY and false for ALLOW_NONE. For ALLOW_AUTHn, it returns the state of the Auth flag maintained in the `FileSystem` class.

Parameters:
access - ACCESS_READ or ACCESS_WRITE
Returns:
true if the specified access is allowed, false otherwise

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacardx.framework.FileSystem

```
java.lang.Object
   |
   +----javacardx.framework.File
           |
           +----javacardx.framework.DedicatedFile
                   |
                   +----javacardx.framework.FileSystem
```

---

public class **FileSystem**
extends DedicatedFile

FileSystem is a subclass of DedicatedFile and it is the "root" DF of the applet. It contains several kinds of methods:

- get and set state values: Auth1 and Auth2 flags and current DF, EF, and record
- find files via name or FID
- handle ISO 7618-4 file-oriented APDUs

Current DF, EF and record number are updated through their setXXX methods and explicit and implicit file selection as defined in ISO 7816. If the current DF is updated, the current EF and the current record number are reset to null and 0 respectively. If the current EF is updated, the current record number is reset to 0 and the current DF points to the parent of the current EF.

Constructor Index

o **FileSystem**(byte)
Constructs an instance of an ISO 7816-4 file system.

Method Index

o **appendRecord**(APDU)
Handles APPEND RECORD command APDU as specified by ISO 7816-4.
o **eraseBinary**(APDU)
Handles ERASE BINARY command APDU as specified by ISO 7816-4.
o **getAuthFlag**(byte)
Get authorization flag.
o **getCurrentDedicatedFile**()
Get current DF.
o **getCurrentElementaryFile**()
Get current EF.
o **getCurrentRecNum**()
Get current record number.
o **getData**(APDU)
Handles GET DATA command APDU as specified by ISO 7816-4.

o **process**(APDU)
  Handles FileSystem APDUs as specified by ISO 7816-4.
o **putData**(APDU)
  Handles PUT DATA command APDU as specified by ISO 7816-4.
o **readBinary**(APDU)
  Handles READ BINARY command APDU as specified by ISO 7816-4.
o **readRecord**(APDU)
  Handles READ RECORD command APDU as specified by ISO 7816-4.
o **reset**()
  Reset the FileSystem internal state.
o **select**(APDU)
  Handles SELECT command APDU as specified by ISO 7816-4.
o **selectFile**(File)
  Make the specified file the current DF or the current EF.
o **setAuthFlag**(byte, boolean)
  Set authorization flag.
o **setCurrentDedicatedFile**(DedicatedFile)
  Set current DF.
o **setCurrentElementaryFile**(ElementaryFile)
  Set current EF.
o **setCurrentRecNum**(byte)
  Set the current record number.
o **updateBinary**(APDU)
  Handles UPDATE BINARY command APDU as specified by ISO 7816-4.
o **updateRecord**(APDU)
  Handles UPDATE RECORD command APDU as specified by ISO 7816-4.
o **writeBinary**(APDU)
  Handles WRITE BINARY command APDU as specified by ISO 7816-4.
o **writeRecord**(APDU)
  Handles WRITE RECORD command APDU as specified by ISO 7816-4.

Constructors

o **FileSystem**

```
public FileSystem(byte maxChildFiles)
```

Constructs an instance of an ISO 7816-4 file system.

Parameters:
maxChildFiles - the maximum number of child files for this DF

Methods

o **reset**

```
public void reset()
```

Reset the FileSystem internal state. This method resets currentDedicatedFile, currentElementaryFile, currentRecordNumber and authorizationFlags to their initial values.

- currentDedicatedFile = this (FileSystem object itself)
- currentElementaryFile = null

- currentRecordNumber = 0 (has no meaning in the context).
- authorizationFlags = false

o **getCurrentDedicatedFile**

```
public DedicatedFile getCurrentDedicatedFile()
```

Get current DF.

Returns:
the current DF

o **setCurrentDedicatedFile**

```
public void setCurrentDedicatedFile(DedicatedFile DF)
```

Set current DF.

Parameters:
DF - set the current DedicatedFile to this DF

o **getCurrentElementaryFile**

```
public ElementaryFile getCurrentElementaryFile()
```

Get current EF.

Returns:
the current EF

o **setCurrentElementaryFile**

```
public void setCurrentElementaryFile(ElementaryFile EF)
```

Set current EF.

Parameters:
EF - set the current ElementaryFile to this EF

o **getCurrentRecNum**

```
public byte getCurrentRecNum()
```

Get current record number.

Returns:
the current record number

o **setCurrentRecNum**

```
public void setCurrentRecNum(byte recNum)
```

Set the current record number.

Parameters:
recNum - set the current record number to recNum

o **getAuthFlag**

```
public boolean getAuthFlag(byte number)
```

Get authorization flag.

Parameters:
number - the number (1 or 2) of the authorization flag
Returns:
the value of the authorization flag

o **setAuthFlag**

```
public void setAuthFlag(byte number,
                        boolean value)
```

Set authorization flag.

Parameters:
number - the number (1 or 2) of the authorization flag
value - the value of the authorization flag

o **selectFile**

```
public void selectFile(File file)
```

Make the specified file the current DF or the current EF.

Parameters:
file - the file reference

o **process**

```
public boolean process(APDU apdu) throws ISOException
```

Handles FileSystem APDUs as specified by ISO 7816-4. This method simply dispatches to other methods in this class based on the INS in the APDU.

Parameters:
apdu - the APDU object
Returns:
true if this method can handle the apdu with normal completion, or false if this method can not handle the APDU (i.e can not recoganize INS in the APDU)
Throws: ISOException
with the resulting SW (other than 0x9000) as defined in ISO 7816

o **select**

```
public void select(APDU apdu) throws ISOException
```

Handles SELECT command APDU as specified by ISO 7816-4.

Parameters:
apdu - the APDU object

Throws: ISOException
If a problem is encountered, throws ISOException with the resulting SW (other than 0x9000) as defined in ISO 7816

o **readBinary**

```
protected void readBinary(APDU apdu) throws ISOException
```

Handles READ BINARY command APDU as specified by ISO 7816-4.

Parameters:
apdu - the APDU object
Throws: ISOException
If a problem is encountered, throws ISOException with the resulting SW (other than 0x9000) as defined in ISO 7816

o **writeBinary**

```
protected void writeBinary(APDU apdu) throws ISOException
```

Handles WRITE BINARY command APDU as specified by ISO 7816-4.

Parameters:
apdu - the APDU object
Throws: ISOException
If a problem is encountered, throws ISOException with the resulting SW (other than 0x9000) as defined in ISO 7816

o **updateBinary**

```
protected void updateBinary(APDU apdu) throws ISOException
```

Handles UPDATE BINARY command APDU as specified by ISO 7816-4.

Parameters:
apdu - the APDU object
Throws: ISOException
If a problem is encountered, throws ISOException with the resulting SW (other than 0x9000) as defined in ISO 7816

o **eraseBinary**

```
protected void eraseBinary(APDU apdu) throws ISOException
```

Handles ERASE BINARY command APDU as specified by ISO 7816-4.

Parameters:
apdu - the APDU object
Throws: ISOException
If a problem is encountered, throws ISOException with the resulting SW (other than 0x9000) as defined in ISO 7816

o **readRecord**

`protected void readRecord(APDU apdu) throws ISOException`

Handles READ RECORD command APDU as specified by ISO 7816-4.

Parameters:
apdu - the APDU object
Throws: ISOException
If a problem is encountered, throws ISOException with the resulting SW (other than 0x9000) as defined in ISO 7816

o **writeRecord**

`protected void writeRecord(APDU apdu) throws ISOException`

Handles WRITE RECORD command APDU as specified by ISO 7816-4.

Parameters:
apdu - the APDU object
Throws: ISOException
If a problem is encountered, throws ISOException with the resulting SW (other than 0x9000) as defined in ISO 7816

o **updateRecord**

`protected void updateRecord(APDU apdu) throws ISOException`

Handles UPDATE RECORD command APDU as specified by ISO 7816-4.

Parameters:
apdu - the APDU object
Throws: ISOException
If a problem is encountered, throws ISOException with the resulting SW (other than 0x9000) as defined in ISO 7816

o **appendRecord**

`protected void appendRecord(APDU apdu) throws ISOException`

Handles APPEND RECORD command APDU as specified by ISO 7816-4.

Parameters:
apdu - the APDU object
Throws: ISOException
If a problem is encountered, throws ISOException with the resulting SW (other than 0x9000) as defined in ISO 7816

o **getData**

`protected void getData(APDU apdu) throws ISOException`

Handles GET DATA command APDU as specified by ISO 7816-4.

Parameters:
apdu - the APDU object

Throws: ISOException
If a problem is encountered, throws ISOException with the resulting SW (other than 0x9000) as defined in ISO 7816

o **putData**

`protected void putData(APDU apdu) throws ISOException`

Handles PUT DATA command APDU as specified by ISO 7816-4.

Parameters:
apdu - the APDU object
Throws: ISOException
If a problem is encountered, throws ISOException with the resulting SW (other than 0x9000) as defined in ISO 7816

All Packages Class Hierarchy This Package Previous Next Index

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacardx.framework.LinearFixedFile

```
java.lang.Object
   |
   +----javacardx.framework.File
           |
           +----javacardx.framework.ElementaryFile
                   |
                   +----javacardx.framework.LinearVariableFile
                           |
                           +----javacardx.framework.LinearFixedFile
```

public class **LinearFixedFile**
extends LinearVariableFile

Linear fixed-length record files.

Constructor Index

o **LinearFixedFile**(short, byte, byte)
    Constructor.

Method Index

o **addRecord**(byte[])
    Add (append) a new record to the file.
o **addRecord**(short)
    Add (append) a new record to the file.

Constructors

o **LinearFixedFile**

```
 public LinearFixedFile(short FID,
                        byte maxNumRecords,
                        byte recordLength)
```

Constructor.

Parameters:
FID - the file's 16-bit FID
maxNumRecords - the maximum number of records in this file
recordLength - the fixed record length for this file

Methods

o **addRecord**

```
 public void addRecord(byte record[]) throws ISOException
```

Add (append) a new record to the file. Note that the record reference is stored in the file object. A copy of the record byte array is not made.

Parameters:
record - the record byte array
Throws: ISOException
if record length is wrong or this file is full.
- ISOException.reason = ISO.SW_WRONG_LENGTH
- ISOException.reason = ISO.SW_FILE_FULL

Overrides:
addRecord in class LinearVariableFile

o **addRecord**

```
 public void addRecord(short length) throws ISOException
```

Add (append) a new record to the file. This creates a new record byte array.

Parameters:
length - the size of the new record byte array to be added
Throws: ISOException
if record length is wrong or this file is full.
- ISOException.reason = ISO.SW_WRONG_LENGTH
- ISOException.reason = ISO.SW_FILE_FULL

Overrides:
addRecord in class LinearVariableFile

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacardx.framework.LinearVariableFile

```
java.lang.Object
   |
   +----javacardx.framework.File
           |
           +----javacardx.framework.ElementaryFile
                   |
                   +----javacardx.framework.LinearVariableFile
```

public class **LinearVariableFile**
extends ElementaryFile

This is the class for all linear variable-length record files, and the base class for linear variable-fixed and cyclic record files.

Variable Index

o **DIRECTION_FIRST**
Direction mode parameter used with findRecord method.
o **DIRECTION_LAST**
Direction mode parameter used with findRecord method.
o **DIRECTION_NEXT**
Direction mode parameter used with findRecord method.
o **DIRECTION_PREV**
Direction mode parameter used with findRecord method.

Constructor Index

o **LinearVariableFile**(short, byte)
Constructor.

Method Index

o **addRecord**(byte[])
Add (append) a new record to the file.
o **addRecord**(short)
Add (append) a new record to the file.
o **findRecord**(byte, byte, byte, byte)
Find the record.
o **getMaxNumRecords**()
Get the maximum number of records in this file.
o **getNumRecords**()
Get the actual number of records in this file.
o **getRecord**(byte)
Get the record byte array for the specified record number.

o **increaseMaxNumRecords**(byte)
Increase the maximum number of records in this file.

Variables

o **DIRECTION_FIRST**

```
public static final byte DIRECTION_FIRST
```

Direction mode parameter used with findRecord method. See findRecord for more details

o **DIRECTION_LAST**

```
public static final byte DIRECTION_LAST
```

Direction mode parameter used with findRecord method. See findRecord for more details

o **DIRECTION_NEXT**

```
public static final byte DIRECTION_NEXT
```

Direction mode parameter used with findRecord method. See findRecord for more details

o **DIRECTION_PREV**

```
public static final byte DIRECTION_PREV
```

Direction mode parameter used with findRecord method. See findRecord for more details

Constructors

o **LinearVariableFile**

```
public LinearVariableFile(short FID,
                          byte maxNumRecords)
```

Constructor.

Parameters:
FID - the file’s 16-bit FID
maxNumRecords - the maximum number of records in this file

Methods

o **getMaxNumRecords**

```
public byte getMaxNumRecords()
```

Get the maximum number of records in this file.

Returns:
maxNumRecords

o **increaseMaxNumRecords**

```
public boolean increaseMaxNumRecords(byte number)
```

Increase the maximum number of records in this file.

Parameters:
number - increase the maximum number of records to this number
Returns:
true if the increase was successful, false otherwise

o **getNumRecords**

```
public byte getNumRecords()
```

Get the actual number of records in this file.

Returns:
numRecords

o **addRecord**

```
public void addRecord(byte record[]) throws ISOException
```

Add (append) a new record to the file. Note that the record reference is stored in the file object. A copy of the record byte array is not made.

Parameters:
record - the record byte array
Throws: ISOException
if the file is full.
- ISOException.reason = ISO.SW_FILE_FULL

o **addRecord**

```
public void addRecord(short length) throws ISOException
```

Add (append) a new record to the file. This creates a new record byte array and sets the array value to 0s.

Parameters:
length - the size of the new record byte array to be added
Throws: ISOException
if the file is full.
- ISOException.reason = ISO.SW_FILE_FULL

o **getRecord**

```
public byte[] getRecord(byte recordNum)
```

Get the record byte array for the specified record number. This is a reference to the actual file data, not a copy of the file data. Records are in the order that they were added to the file. Record number is in the range from 1 to the number of records in the file

Parameters:
recordNum - the index (first record = 1) of the record.
Returns:
record or null if the record is not found

o **findRecord**

```
public byte findRecord(byte direction,
                       byte currentRecNumber,
                       byte firstByte,
                       byte secondByte)
```

Find the record. Using the specified direction and current record number as the starting point, find the record for which first and second byte match `firstByte` and `secondByte` specified in the parameter . Records are numbered in the order that they were added to the file. (See Annex C of ISO 7816-4 for details)

DIRECTION_xxx constants.
- DIRECTION_FIRST: Start at the first record in file
- DIRECTION_LAST: Start at the last record in file
- DIRECTION_NEXT: Start at the current record and move forward
- DIRECTION_PREV: Start at the current record and move backward

Parameters:
direction - one of the DIRECTION_xxx constants.
firstByte - if non-0, the record’s first byte must match this value; if 0, any value of the record’s first byte matches.
secondByte - if non-0, the record’s second byte must match this value; if 0, any value of the record’s second byte matches.
currentRecNumber - current record number. If 0, the current record is undefined.
Returns:
the record number, or 0 if the record is not found

All Packages Class Hierarchy This Package Previous Next Index

Class javacardx.framework.TransparentFile

```
java.lang.Object
   |
   +----javacardx.framework.File
           |
           +----javacardx.framework.ElementaryFile
                   |
                   +----javacardx.framework.TransparentFile
```

public class **TransparentFile**
extends ElementaryFile

This is the class for all transparent files. Data is stored in the file as a sequence of data units.

Constructor Index

o **TransparentFile**(short, byte[])
  Constructor, with data byte array specified.
o **TransparentFile**(short, short)
  Constructor, with data byte array size specified.

Method Index

o **getData**()
  Gets the byte array containing the data for this file.

Constructors

o **TransparentFile**

```
public TransparentFile(short FID,
                       byte data[])
```

Constructor, with data byte array specified. Note that the data reference is stored in the file object. A copy of the data byte array is not made.

Parameters:
FID - the file’s 16-bit FID
data - the data byte array of this file

o **TransparentFile**

```
public TransparentFile(short FID,
                       short length)
```

Constructor, with data byte array size specified. This creates a new data byte arra and set the array value to 0s.

Parameters:
FID - the file’s 16-bit FID
length - the length of the data byte array

Methods

o **getData**

```
public byte[] getData()
```

Gets the byte array containing the data for this file. This is a reference to the actual file data, not a copy of the file data.

Returns:
data stored in the file

`All Packages  Class Hierarchy  Index`

package javacardx.crypto

Class Index

- AsymKey
- DES3_Key
- DES_Key
- Key
- MessageDigest
- PrivateKey
- PublicKey
- RSA_CRT_PrivateKey
- RSA_PrivateKey
- RSA_PublicKey
- RandomData
- Sha1MessageDigest
- SymKey

Exception Index

- CryptoException

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacardx.crypto.AsymKey

```
java.lang.Object
   |
   +----javacardx.crypto.Key
           |
           +----javacardx.crypto.AsymKey
```

public abstract class **AsymKey**
extends Key

The AsymKey class is the base class for keys used in asymmetric algorithms.

Constructor Index

o **AsymKey**(short)
Constructs an asymmetric key with a specific bit length

Method Index

o **getBitLength**()
Gets the length of the key in bits.
o **isSupportedLength**(short)
Reports if the implementation supports the requested key length (length in bits).

Constructors

o **AsymKey**

```
public AsymKey(short length)
```

Constructs an asymmetric key with a specific bit length

Parameters:
length - the length of the key in bits

Methods

o **getBitLength**

```
public final short getBitLength()
```

Gets the length of the key in bits.

Returns:
the length of the key in bits

o **isSupportedLength**

```
public static boolean isSupportedLength(short length)
```

Reports if the implementation supports the requested key length (length in bits).

Parameters:
length - the length of bits that is being requested.

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacardx.crypto.CryptoException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.lang.RuntimeException
                           |
                           +----javacardx.crypto.CryptoException
```

---

public class **CryptoException**
extends RuntimeException

`CryptoException` represents a cryptography-related exception.

Table CryptoException

Reason	**Description**
GENERAL	general cryptographic exception
MD_GEN	message digest generation failed
UNINIT_KEY	use of uninitialized key
INVALID_PARAM	invalid parameter passed to a method
ENC_NOT_SUPPORTED	encryption is not supported

Variable Index

o **ENC_NOT_SUPPORTED**
o **GENERAL**
o **INVALID_PARAM**
o **MD_GEN**
o **UNINIT_KEY**

Constructor Index

o **CryptoException**(short)
Constructs a CryptoException with the specified reason.

Variables

o **GENERAL**

```
public static final short GENERAL
```

o **MD_GEN**

```
public static final short MD_GEN
```

o **UNINIT_KEY**

```
public static final short UNINIT_KEY
```

o **INVALID_PARAM**

```
public static final short INVALID_PARAM
```

o **ENC_NOT_SUPPORTED**

```
public static final short ENC_NOT_SUPPORTED
```

Constructors

o **CryptoException**

```
public CryptoException(short reason)
```

Constructs a CryptoException with the specified reason.

Parameters:
reason - the reason for the exception.

Class javacardx.crypto.DES3_Key

```
java.lang.Object
   |
   +----javacardx.crypto.Key
           |
           +----javacardx.crypto.SymKey
                   |
                   +----javacardx.crypto.DES3_Key
```

public class **DES3_Key**
extends SymKey

DES3_Key contains an 16 byte key for triple DES operations in either ECB or CBC mode.

DES operates on a block size of 8 bytes and all input parameters to these methods are expected to be multiples of 8 bytes. In each case the caller is responisble for padding the input.

Note: this class does not implement encryption functionality.

See Also:
DES_Key, DES_EncKey, DES3_EncKey

Constructor Index

o **DES3_Key**()
Creates a key for triple DES operation with a block size of 8 bytes and a key length of 16 bytes.

Method Index

o **decryptCBC**(byte[], short, short, byte[], short)
Decrypts data using triple DES in CBC mode.
o **decryptECB**(byte[], short, short, byte[], short)
Decrypts data using triple DES in ECB mode.
o **generateMAC**(byte[], short, short, byte[], short, byte)
Generates a MAC using triple DES decryption in CBC mode.
o **verifyMAC**(byte[], short, byte, byte[], short, short)
Verifies a MAC on signed data using triple DES decryption in CBC mode.

Constructors

o **DES3_Key**

```
public DES3_Key()
```

Creates a key for triple DES operation with a block size of 8 bytes and a key length of 16 bytes.

Methods

o **decryptECB**

```
public void decryptECB(byte inBuff[],
                       short inOffset,
                       short inLength,
                       byte outBuff[],
                       short outOffset)
```

Decrypts data using triple DES in ECB mode.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin decryption
inLength - the length to decrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer
Overrides:
decryptECB in class SymKey

o **decryptCBC**

```
public void decryptCBC(byte inBuff[],
                       short inOffset,
                       short inLength,
                       byte outBuff[],
                       short outOffset)
```

Decrypts data using triple DES in CBC mode.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin decryption
inLength - the length to decrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer
Overrides:
decryptCBC in class SymKey

o **generateMAC**

```
public void generateMAC(byte inBuff[],
                        short inOffset,
                        short inLength,
                        byte outBuff[],
                        short outOffset,
                        byte length)
```

Generates a MAC using triple DES decryption in CBC mode.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin MAC generation

inLength - the length to encrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer
outLength - the length of the MAC to generate
Overrides:
generateMAC in class SymKey

o **verifyMAC**

```
public boolean verifyMAC(byte macBuffer[],
                         short macOffset,
                         byte macLength,
                         byte inData[],
                         short inOffset,
                         short inLength)
```

Verifies a MAC on signed data using triple DES decryption in CBC mode.

Parameters:
macBuffer - the buffer containing the MAC to verify.
macOffset - the offset into the MAC buffer
macLength - the length of the MAC
inData - the buffer containing the input data.
inOffset - the offset into the input data buffer
inLength - the length of the input data buffer
Returns:
true if the data if the given MAC is verified, false otherwise.
Overrides:
verifyMAC in class SymKey

All Packages Class Hierarchy This Package Previous Next Index

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacardx.crypto.DES_Key

```
java.lang.Object
   |
   +----javacardx.crypto.Key
           |
           +----javacardx.crypto.SymKey
                   |
                   +----javacardx.crypto.DES_Key
```

public class **DES_Key**
extends SymKey

DES_Key contains an 8 byte key for single DES operations in either ECB or CBC mode.

DES operates on a block size of 8 bytes and all input parameters to these methods are expected to be multiples of 8 bytes. In each case the caller is responisble for padding the input.

Note: this class does not implement encryption functionality.

See Also:
DES3_Key, DES_EncKey, DES3_EncKey

Constructor Index

o **DES_Key**()
Creates a key for single DES operation with a block size of 8 bytes and a key length of 8 bytes.

Method Index

o **decryptCBC**(byte[], short, short, byte[], short)
Decrypts data using single DES in CBC mode.
o **decryptECB**(byte[], short, short, byte[], short)
Decrypts data using single DES in ECB mode.
o **generateMAC**(byte[], short, short, byte[], short, byte)
Generates a MAC using single DES decryption in CBC mode.
o **verifyMAC**(byte[], short, byte, byte[], short, short)
Verifies a MAC on signed data using single DES decryption in CBC mode.

Constructors

o **DES_Key**

```
public DES_Key()
```

Creates a key for single DES operation with a block size of 8 bytes and a key length of 8 bytes.

Methods

o **decryptECB**

```
public void decryptECB(byte inBuff[],
                       short inOffset,
                       short inLength,
                       byte outBuff[],
                       short outOffset)
```

Decrypts data using single DES in ECB mode.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin decryption
inLength - the length to decrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer
Overrides:
decryptECB in class SymKey

o **decryptCBC**

```
public void decryptCBC(byte inBuff[],
                       short inOffset,
                       short inLength,
                       byte outBuff[],
                       short outOffset)
```

Decrypts data using single DES in CBC mode.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin decryption
inLength - the length to decrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer
Overrides:
decryptCBC in class SymKey

o **generateMAC**

```
public void generateMAC(byte inBuff[],
                        short inOffset,
                        short inLength,
                        byte outBuff[],
                        short outOffset,
                        byte length)
```

Generates a MAC using single DES decryption in CBC mode.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin encryption

inLength - the length to encrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer
outLength - the length of the MAC to generate

Overrides:
generateMAC in class SymKey

o **verifyMAC**

```
 public boolean verifyMAC(byte macBuffer[],
                          short macOffset,
                          byte macLength,
                          byte inData[],
                          short inOffset,
                          short inLength)
```

Verifies a MAC on signed data using single DES decryption in CBC mode.

Parameters:
macBuffer - the buffer containing the MAC to verify.
macOffset - the offset into the MAC buffer
macLength - the length of the MAC
inData - the buffer containing the input data.
inOffset - the offset into the input data buffer
inLength - the length of the input data buffer

Returns:
true if the data if the given MAC is verified, false otherwise.

Overrides:
verifyMAC in class SymKey

All Packages Class Hierarchy This Package Previous Next Index

All Packages Class Hierarchy This Package Previous Next Index

Class javacardx.crypto.Key

```
java.lang.Object
   |
   +----javacardx.crypto.Key
```

public abstract class **Key**
extends Object

The Key class is the base class for keys.

Constructor Index

o **Key**()
Constructs a key.

Method Index

o **clearKey**()
Clears the key and sets its initialized state to false.
o **isInitialized**()
Reports the initialized state of the key.

Constructors

o **Key**

```
 public Key()
```

Constructs a key.

Methods

o **isInitialized**

```
 public boolean isInitialized()
```

Reports the initialized state of the key. Keys must be initialized before being used.

Returns:
true if the key has been initialized.

o **clearKey**

```
 public void clearKey()
```

Clears the key and sets its initialized state to false.

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacardx.crypto.MessageDigest

```
java.lang.Object
   |
   +----javacardx.crypto.MessageDigest
```

---

public class **MessageDigest**
extends Object

The MessageDigest class is the base class for hashing algorthims.

Constructor Index

o **MessageDigest**(short, short)
    Creates a message digest with a given block size and hash result size.

Method Index

o **blockSize**()
    Gets the block size in bytes.
o **generateDigest**(byte[], short, short, byte[], short)
    generates a hash of the input data.
o **hashSize**()
    Gets the hash size in bytes.

Constructors

o **MessageDigest**

```
 public MessageDigest(short blockSize,
                      short hashSize)
```

Creates a message digest with a given block size and hash result size.

Parameters:
blockSize - the size in bytes of the blocks processed
hashSize - the size in bytes of the resulting hash value

Methods

o **blockSize**

```
 public short blockSize()
```

Gets the block size in bytes.

Returns:
the block size in bytes

o **hashSize**

```
public short hashSize()
```

Gets the hash size in bytes.

Returns:
the hash size in bytes

o **generateDigest**

```
public void generateDigest(byte inBuff[],
                           short inOffset,
                           short inLength,
                           byte outBuff[],
                           short outOffset)
```

generates a hash of the input data.

Parameters:
inBuff - the input buffer of data to be hashed
inOffset - the offset into the input buffer at which to begin hash generation
inLength - the length to hash
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer where the resulting hash value begins

Class javacardx.crypto.PrivateKey

```
java.lang.Object
   |
   +----javacardx.crypto.Key
           |
           +----javacardx.crypto.AsymKey
                   |
                   +----javacardx.crypto.PrivateKey
```

public abstract class **PrivateKey**
extends AsymKey

The PrivateKey class is the base class for private keys used in asymmetric algorithms.

Constructor Index

o **PrivateKey**(short)
Creates a private key with a specific bit length.

Method Index

o **sign**(byte[], short, short, byte[], short)
Signs data using this key.

Constructors

o **PrivateKey**

```
public PrivateKey(short length)
```

Creates a private key with a specific bit length.

Parameters:
length - the length in bits

Methods

o **sign**

```
public abstract void sign(byte inBuff[],
                          short inOffset,
                          short inLength,
                          byte outBuff[],
                          short outOffset)
```

Signs data using this key.

Parameters:
inBuff - the input buffer containing data to be signed
inOffset - the offset into the input buffer
inLength - the length
outBuff - the output buffer, may be the same as the input buffer; contains the resulting signature
outOffset - the offset into the output buffer

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacardx.crypto.PublicKey

```
java.lang.Object
   |
   +----javacardx.crypto.Key
           |
           +----javacardx.crypto.AsymKey
                   |
                   +----javacardx.crypto.PublicKey
```

---

public abstract class **PublicKey**
extends AsymKey

The PublicKey class is the base class for public keys used in asymmetric algorithms.

Constructor Index

o **PublicKey**(short)
Creates a public key with a specific bit length.

Method Index

o **verify**(byte[], short, short, byte[], short, short)
Verifies signed data using this key.

Constructors

o **PublicKey**

```
public PublicKey(short length)
```

Creates a public key with a specific bit length.

Parameters:
length - the length in bits

Methods

o **verify**

```
public abstract boolean verify(byte msgDigest[],
                               short msgOffset,
                               short msgLength,
                               byte signedData[],
                               short signOffset,
                               short signLength)
```

Verifies signed data using this key.

Parameters:
msgDigest - the buffer containing the hash result.
msgOffset - the offset into the hash result buffer
msgLength - the length of the hash
signedData - the buffer containing the signed data.
signOffset - the offset into the signed data buffer
signLength - the of the signed data buffer
Returns:
true if the data is properly signed.

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacardx.crypto.RSA_CRT_PrivateKey

```
java.lang.Object
   |
   +----javacardx.crypto.Key
           |
           +----javacardx.crypto.AsymKey
                   |
                   +----javacardx.crypto.PrivateKey
                           |
                           +----javacardx.crypto.RSA_CRT_PrivateKey
```

---

public class **RSA_CRT_PrivateKey**
extends PrivateKey

The RSA_CRT_PrivateKey class is used to sign data using the RSA algorithm in its Chinese Remainder Theorem form.

Let $S = m^d \bmod n$, where m is the data to be signed, d is the private key exponent, and n is private key modulus composed of two prime numbers p and q. The following names are used in the setter methods in this class:

P, the prime factor p
Q, the prime factor q.
PQ = $p^{-1} \bmod q$
DP1 = $d \bmod (p - 1)$
DQ1 = $d \bmod (q - 1)$

See Also:
RSA_Key

Constructor Index

o **RSA_CRT_PrivateKey**(short)
Constructs a key with a specific bit length

Method Index

o **isInitialized**()
Reports the initialized state of the key.
o **setDP1**(byte[], short, short)
Sets the value of the DP1 parameter.
o **setDQ1**(byte[], short, short)
Sets the value of the DQ1 key.
o **setP**(byte[], short, short)
Sets the value of the P parameter.

o **setPQ**(byte[], short, short)
  Sets the value of the PQ parameter.
o **setQ**(byte[], short, short)
  Sets the value of the Q parameter.
o **sign**(byte[], short, short, byte[], short)
  Signs data using this key.

Constructors

o **RSA_CRT_PrivateKey**

```
public RSA_CRT_PrivateKey(short length)
```

Constructs a key with a specific bit length

Parameters:
  length - the length of the key in bits

Methods

o **isInitialized**

```
public boolean isInitialized()
```

Reports the initialized state of the key. All five CRT parameter must be initialized before the key can be used.

Returns:
  true if the key has been initialized.
Overrides:
  isInitialized in class Key

o **setP**

```
public void setP(byte buffer[],
                 short offset,
                 short length)
```

Sets the value of the P parameter.

Parameters:
  buffer - the input buffer
  offset - the offset into the input buffer at which the parameter value begins
  length - the length of the parameter

o **setQ**

```
public void setQ(byte buffer[],
                 short offset,
                 short length)
```

Sets the value of the Q parameter.

Parameters:
  buffer - the input buffer
  offset - the offset into the input buffer at which the parameter value begins
  length - the length of the parameter

o **setDP1**

```
public void setDP1(byte buffer[],
                   short offset,
                   short length)
```

Sets the value of the DP1 parameter.

Parameters:
  buffer - the input buffer
  offset - the offset into the input buffer at which the parameter value begins
  length - the length of the parameter

o **setDQ1**

```
public void setDQ1(byte buffer[],
                   short offset,
                   short length)
```

Sets the value of the DQ1 key.

Parameters:
  buffer - the input buffer
  offset - the offset into the input buffer at which the parameter value begins
  length - the length of the parameter

o **setPQ**

```
public void setPQ(byte buffer[],
                  short offset,
                  short length)
```

Sets the value of the PQ parameter.

Parameters:
  buffer - the input buffer
  offset - the offset into the input buffer at which the parameter value begins
  length - the length of the parameter

o **sign**

```
public void sign(byte inBuff[],
                 short inOffset,
                 short inLength,
                 byte outBuff[],
                 short outOffset)
```

Signs data using this key.

Parameters:
  inBuff - the input buffer containing data to be signed
  inOffset - the offset into the input buffer

inLength - the length
outBuff - the output buffer, may be the same as the input buffer; contains the resulting signature
outOffset - the offset into the output buffer

Overrides:
sign in class PrivateKey

All Packages Class Hierarchy This Package Previous Next Index

All Packages Class Hierarchy This Package Previous Next Index

---

Class javacardx.crypto.RSA_PrivateKey

```
java.lang.Object
   |
   +----javacardx.crypto.Key
           |
           +----javacardx.crypto.AsymKey
                   |
                   +----javacardx.crypto.PrivateKey
                           |
                           +----javacardx.crypto.RSA_PrivateKey
```

---

public class **RSA_PrivateKey**
extends PrivateKey

The RSA_PrivateKey class is used to sign data using the RSA algorithm in its modulus/exponent form.

See Also:
RSA_PublicKey, RSA_CRT_PrivateKey

Constructor Index

o **RSA_PrivateKey**(short)
Constructs a key with a specific bit length

Method Index

o **isInitialized**()
Reports the initialized state of the key.
o **setExponent**(byte[], short, short)
Sets the exponent value of the key.
o **setModulus**(byte[], short, short)
Sets the modulus value of the key.
o **sign**(byte[], short, short, byte[], short)
Signs data using this key.

Constructors

o **RSA_PrivateKey**

```
public RSA_PrivateKey(short length)
```

Constructs a key with a specific bit length

Parameters:
length - the length of the key in bits

Methods

o **isInitialized**

```
public boolean isInitialized()
```

Reports the initialized state of the key. All five CRT parameter must be initialized before the key can be used.

Returns:
true if the key has been initialized.
Overrides:
isInitialized in class Key

o **setModulus**

```
public void setModulus(byte buffer[],
                       short offset,
                       short length)
```

Sets the modulus value of the key. When both the modulus and exponent are set the key is initialized and ready for use.

Parameters:
buffer - the input buffer
offset - the offset into the input buffer at which modulus value begins
length - the length of the modulus

o **setExponent**

```
public void setExponent(byte buffer[],
                        short offset,
                        short length)
```

Sets the exponent value of the key. When both the modulus and exponent are set the key is initialized and ready for use.

Parameters:
buffer - the input buffer
offset - the offset into the input buffer at which the exponent value begins
length - the length of the exponent

o **sign**

```
public void sign(byte inBuff[],
                 short inOffset,
                 short inLength,
                 byte outBuff[],
                 short outOffset)
```

Signs data using this key.

Parameters:
inBuff - the input buffer containing data to be signed
inOffset - the offset into the input buffer
inLength - the length

outBuff - the output buffer, may be the same as the input buffer; contains the resulting signature
outOffset - the offset into the output buffer
Overrides:
sign in class PrivateKey

All Packages Class Hierarchy This Package Previous Next Index

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacardx.crypto.RSA_PublicKey

```
java.lang.Object
   |
   +----javacardx.crypto.Key
           |
           +----javacardx.crypto.AsymKey
                   |
                   +----javacardx.crypto.PublicKey
                           |
                           +----javacardx.crypto.RSA_PublicKey
```

public class **RSA_PublicKey**
extends PublicKey

The RSA_PublicKey is used to verify signatures on signed data using the RSA algorithm in its modulus/exponent form.

See Also:
RSA_CRT_Key

Constructor Index

o **RSA_PublicKey**(short)
Creates an empty key with a specific bit length.

Method Index

o **isInitialized**()
Reports the initialized state of the key.
o **setExponent**(byte[], short, short)
Sets the exponent value of the key.
o **setModulus**(byte[], short, short)
Sets the modulus value of the key.
o **verify**(byte[], short, short, byte[], short, short)
Verifies signed data using this key.

Constructors

o **RSA_PublicKey**

```
public RSA_PublicKey(short length)
```

Creates an empty key with a specific bit length.

Parameters:
length - the length in bits

Methods

o **isInitialized**

```
public boolean isInitialized()
```

Reports the initialized state of the key. Both the modulus and exponent must be initialized before the key can be used.

Returns:
true if the key has been initialized.
Overrides:
isInitialized in class Key

o **setModulus**

```
public void setModulus(byte buffer[],
                       short offset,
                       short length)
```

Sets the modulus value of the key. When both the modulus and exponent are set the key is initialized and ready for use.

Parameters:
buffer - the input buffer
offset - the offset into the input buffer at which modulus value begins
length - the length of the modulus

o **setExponent**

```
public void setExponent(byte buffer[],
                        short offset,
                        short length)
```

Sets the exponent value of the key. When both the modulus and exponent are set the key is initialized and ready for use.

Parameters:
buffer - the input buffer
offset - the offset into the input buffer at which the exponent value begins
length - the length of the exponent

o **verify**

```
public boolean verify(byte msgDigest[],
                      short msgOffset,
                      short msgLength,
                      byte signedData[],
                      short signOffset,
                      short signLength)
```

Verifies signed data using this key.

Parameters:
msgDigest - the buffer containing the hash result.
msgOffset - the offset into the hash result buffer

msgLength - the length of the hash
signedData - the buffer containing the signed data.
signOffset - the offset into the signed data buffer
signLength - the of the signed data buffer

Returns:
true if the data is properly signed.

Overrides:
verify in class PublicKey

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacardx.crypto.RandomData

```
java.lang.Object
   |
   +----javacardx.crypto.RandomData
```

---

public class **RandomData**
extends Object

The RandomData class provides a source of (psuedo) randomness.

Constructor Index

o **RandomData**()

Method Index

o **generateData**(byte[], short, short)
Generates random data.
o **setSeed**(byte[], short, short)
Seeds the random data generator.

Constructors

o **RandomData**

```
public RandomData()
```

Methods

o **generateData**

```
public static void generateData(byte buffer[],
                                short offset,
                                short length)
```

Generates random data.

Parameters:
buffer - the output buffer
offset - the offset into the output buffer
length - the length of random data to generate

o **setSeed**

```
public static void setSeed(byte buffer[],
                           short offset,
                           short length)
```

Seeds the random data generator.

Parameters:
buffer - the input buffer
offset - the offset into the input buffer
length - the length of the seed data

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

`All Packages  Class Hierarchy  This Package  Previous  Next  Index`

Class javacardx.crypto.Sha1MessageDigest

```
java.lang.Object
   |
   +----javacardx.crypto.MessageDigest
           |
           +----javacardx.crypto.Sha1MessageDigest
```

---

public class **Sha1MessageDigest**
extends MessageDigest

The Sha1MessageDigest class implements the SHA1 algorithm.

Constructor Index

o **Sha1MessageDigest**()
Creates a Sha1MessageDigest object with a block size of 64 bytes and a resulting hash value size of 20 bytes.

Method Index

o **generateDigest**(byte[], short, short, byte[], short)
generates a hash of the input data using the SHA1 algorithm.

Constructors

o **Sha1MessageDigest**

```
 public Sha1MessageDigest()
```

Creates a Sha1MessageDigest object with a block size of 64 bytes and a resulting hash value size of 20 bytes.

Methods

o **generateDigest**

```
 public void generateDigest(byte inBuff[],
                            short inOffset,
                            short inLength,
                            byte outBuff[],
                            short outOffset)
```

generates a hash of the input data using the SHA1 algorithm.

Parameters:
inBuff - the input buffer of data to be hashed
inOffset - the offset into the input buffer at which to begin hash generation

inLength - the length to hash
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer where the resulting hash value begins

Overrides:

generateDigest in class MessageDigest

All Packages Class Hierarchy This Package Previous Next Index

All Packages Class Hierarchy This Package Previous Next Index

Class javacardx.crypto.SymKey

```
java.lang.Object
   |
   +----javacardx.crypto.Key
           |
           +----javacardx.crypto.SymKey
```

public abstract class **SymKey**
extends Key

The SymKey class is the base class for keys used in symetric alogrightms (e.g. DES). A key in this class supports two modes of operation, ECB and CBC, and can be used to generate and verify MACs as well as decrypt and optionally encrypt.

Constructor Index

o **SymKey**(short, short)
Constructs a symmetric key object of known block size and key size.

Method Index

o **clearICV**()
Clears the initial chaining vector used in CBC mode operations.
o **decryptCBC**(byte[], short, short, byte[], short)
Decrypts data using this key in CBC mode.
o **decryptECB**(byte[], short, short, byte[], short)
Decrypts data using this key in ECB mode.
o **encryptCBC**(byte[], short, short, byte[], short)
Encrypts data using this key in CBC mode.
o **encryptECB**(byte[], short, short, byte[], short)
Encrypts data using this key in ECB mode.
o **generateMAC**(byte[], short, short, byte[], short, byte)
Generates a MAC using decryption in CBC mode.
o **getBlockSize**()
Gets the block size used by the algorithm associated with this key.
o **getKeyLength**()
Gets the length of the key.
o **setICV**(byte[], short)
Sets the initial chaining vector used in CBC mode operations.
o **setKey**(byte[], short)
Initializes a key from raw key data bytes.
o **verifyMAC**(byte[], short, byte, byte[], short, short)
Verifies signed data using decryption in CBC mode.

Constructors

o **SymKey**

```
public SymKey(short theBlocksize,
              short theKeyLength)
```

Constructs a symmetric key object of known block size and key size.

Parameters:
theBlocksize - the size in bytes of the blocks of data processed by the symmetric key algorithm.
theKeyLength - the size in bytes of the key data

Methods

o **setKey**

```
public void setKey(byte buff[],
                   short offset)
```

Initializes a key from raw key data bytes. After initialization `isInitialized()` returns true. The length of the data in `buff` is the equal to `keyLength()`.

Parameters:
buff - the input buffer
offset - the offset into the input buffer at which the key data begins

o **getBlockSize**

```
public short getBlockSize()
```

Gets the block size used by the algorithm associated with this key.

Returns:
the block size in bytes

o **getKeyLength**

```
public short getKeyLength()
```

Gets the length of the key.

Returns:
the key length in bytes

o **setICV**

```
public void setICV(byte buff[],
                   short offset)
```

Sets the initial chaining vector used in CBC mode operations. The ICV is one block size (`blockSize()`) in length.

Parameters:
buff - the input buffer
offset - the offset into the input buffer at which the ICV begins

o **clearICV**

```
public void clearICV()
```

Clears the initial chaining vector used in CBC mode operations.

o **encryptECB**

```
public void encryptECB(byte inBuff[],
                       short inOffset,
                       short inLength,
                       byte outBuff[],
                       short outOffset)
```

Encrypts data using this key in ECB mode. Not all subclasses will implement this method (in order to avoid import/export restrictions); the default implementation throws a CryptoException with the reason ENC_NOT_SUPPORTED.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin encryption
inLength - the length to encrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer

o **encryptCBC**

```
public void encryptCBC(byte inBuff[],
                       short inOffset,
                       short inLength,
                       byte outBuff[],
                       short outOffset)
```

Encrypts data using this key in CBC mode. Not all subclasses will implement this method (in order to avoid import/export restrictions); the default implementation throws a CryptoException with the reason ENC_NOT_SUPPORTED.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin encryption
inLength - the length to encrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer

o **decryptECB**

```
public abstract void decryptECB(byte inBuff[],
                                short inOffset,
                                short inLength,
                                byte outBuff[],
                                short outOffset)
```

Decrypts data using this key in ECB mode.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin decryption
inLength - the length to decrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer

o **decryptCBC**

```
public abstract void decryptCBC(byte inBuff[],
                                short inOffset,
                                short inLength,
                                byte outBuff[],
                                short outOffset)
```

Decrypts data using this key in CBC mode.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin decryption
inLength - the length to decrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer

o **generateMAC**

```
public abstract void generateMAC(byte inBuff[],
                                 short inOffset,
                                 short inLength,
                                 byte outBuff[],
                                 short outOffset,
                                 byte outLength)
```

Generates a MAC using decryption in CBC mode.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin encryption
inLength - the length to encrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer
outLength - the length of the MAC to generate

o **verifyMAC**

```
public abstract boolean verifyMAC(byte macBuffer[],
                                  short macOffset,
                                  byte macLength,
                                  byte inData[],
                                  short inOffset,
                                  short inLength)
```

Verifies signed data using decryption in CBC mode.

Parameters:
macBuffer - the buffer containing the MAC to verify.
macOffset - the offset into the MAC buffer
macLength - the length of the MAC
inData - the buffer containing the input data.
inOffset - the offset into the input data buffer
inLength - the length of the input data buffer
Returns:
true if the data if the given MAC is verified, false otherwise.

All Packages Class Hierarchy This Package Previous Next Index

package javacardx.cryptoEnc

Class Index

- DES3_EncKey
- DES_EncKey

Class javacardx.cryptoEnc.DES3_EncKey

```
java.lang.Object
   |
   +----javacardx.crypto.Key
           |
           +----javacardx.crypto.SymKey
                   |
                   +----javacardx.crypto.DES3_Key
                           |
                           +----javacardx.cryptoEnc.DES3_EncKey
```

public class **DES3_EncKey**
extends DES3_Key

DES3_EncKey extends DES3_Key by adding encryption functionality

DES operates on a block size of 8 bytes and all input parameters to these methods are expected to be multiples of 8 bytes. In each case the caller is responisble for padding the input.

See Also:
DES_Key, DES_EncKey, DES3_Key

Constructor Index

o **DES3_EncKey**()
Creates a key for triple DES operation with a block size of 8 bytes and a key length of 16 bytes.

Method Index

o **encryptCBC**(byte[], short, short, byte[], short)
Encrypts data using this key with triple DES in CBC mode.
o **encryptECB**(byte[], short, short, byte[], short)
Encrypts data using this key with triple DES in ECB mode.

Constructors

o **DES3_EncKey**

```
public DES3_EncKey()
```

Creates a key for triple DES operation with a block size of 8 bytes and a key length of 16 bytes.

Methods

o **encryptECB**

```
 public void encryptECB(byte inBuff[],
                        short inOffset,
                        short inLength,
                        byte outBuff[],
                        short outOffset)
```

Encrypts data using this key with triple DES in ECB mode.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin encryption
inLength - the length to encrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer
Overrides:
encryptECB in class SymKey

o **encryptCBC**

```
 public void encryptCBC(byte inBuff[],
                        short inOffset,
                        short inLength,
                        byte outBuff[],
                        short outOffset)
```

Encrypts data using this key with triple DES in CBC mode.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin encryption
inLength - the length to encrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer
Overrides:
encryptCBC in class SymKey

Class javacardx.cryptoEnc.DES_EncKey

```
java.lang.Object
   |
   +----javacardx.crypto.Key
           |
           +----javacardx.crypto.SymKey
                   |
                   +----javacardx.crypto.DES_Key
                           |
                           +----javacardx.cryptoEnc.DES_EncKey
```

public class **DES_EncKey**
extends DES_Key

DES_EncKey extends DES_Key by adding encryption functionality.

DES operates on a block size of 8 bytes and all input parameters to these methods are expected to be multiples of 8 bytes. In each case the caller is responisble for padding the input.

See Also:
DES_Key, DES3_Key, DES3_EncKey

Constructor Index

o **DES_EncKey**()
Creates a key for single DES operation with a block size of 8 bytes and a key length of 8 bytes.

Method Index

o **encryptCBC**(byte[], short, short, byte[], short)
Encrypts data using this key with single DES in CBC mode.
o **encryptECB**(byte[], short, short, byte[], short)
Encrypts data using this key with single DES in ECB mode.

Constructors

o **DES_EncKey**

```
 public DES_EncKey()
```

Creates a key for single DES operation with a block size of 8 bytes and a key length of 8 bytes.

Methods

o **encryptECB**

```
 public void encryptECB(byte inBuff[],
                        short inOffset,
                        short inLength,
                        byte outBuff[],
                        short outOffset)
```

Encrypts data using this key with single DES in ECB mode.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin encryption
inLength - the length to encrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer
Overrides:
encryptECB in class SymKey

o **encryptCBC**

```
 public void encryptCBC(byte inBuff[],
                        short inOffset,
                        short inLength,
                        byte outBuff[],
                        short outOffset)
```

Encrypts data using this key with single DES in CBC mode.

Parameters:
inBuff - the input buffer
inOffset - the offset into the input buffer at which to begin encryption
inLength - the length to encrypt
outBuff - the output buffer, may be the same as the input buffer
outOffset - the offset into the output buffer
Overrides:
encryptCBC in class SymKey

All Packages Class Hierarchy This Package Previous Next Index

All Packages Class Hierarchy

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Index of all Fields and Methods

encryptECB(byte[], short, short, byte[], short). Method in class javacardx.cryptoEnc.DES3_EncKey
Encrypts data using this key with triple DES in ECB mode.
encryptECB(byte[], short, short, byte[], short). Method in class javacardx.cryptoEnc.DES_EncKey
Encrypts data using this key with single DES in ECB mode.
encryptECB(byte[], short, short, byte[], short). Method in class javacardx.crypto.SymKey
Encrypts data using this key in ECB mode.
equals(Object). Method in class java.lang.Object
Compares two Objects for equality.
eraseBinary(APDU). Method in class javacardx.framework.FileSystem
Handles ERASE BINARY command APDU as specified by ISO 7816-4.
Exception(). Constructor for class java.lang.Exception
Constructs an Exception instance with reason = 0.
Exception(short). Constructor for class java.lang.Exception
Constructs an Exception instance with the specified reason.

F

FileSystem(byte). Constructor for class javacardx.framework.FileSystem
Constructs an instance of an ISO 7816-4 file system.
FIND_ANY. Static variable in class javacardx.framework.DedicatedFile
Selection mode parameter used with the findFile method.
FIND_CHILD. Static variable in class javacardx.framework.DedicatedFile
Selection mode parameter used with the findFile method.
FIND_CHILD_DF. Static variable in class javacardx.framework.DedicatedFile
Selection mode parameter used with the findFile method.
FIND_CHILD_EF. Static variable in class javacardx.framework.DedicatedFile
Selection mode parameter used with the findFile method.
findDedicatedFile(byte[], short, byte). Method in class javacardx.framework.DedicatedFile
Under this DF, find the DF with the specified name.
findElementaryFile(byte). Method in class javacardx.framework.DedicatedFile
Under this DF, find the EF with the specified SFI.
findFile(byte, short). Method in class javacardx.framework.DedicatedFile
According to the findType, find the file with the specified FID.
findRecord(byte, byte, byte, byte). Method in class javacardx.framework.CyclicFile
Find the record.
findRecord(byte, byte, byte, byte). Method in class javacardx.framework.LinearVariableFile
Find the record.

G

GENERAL. Static variable in class javacardx.crypto.CryptoException
generateData(byte[], short, short). Static method in class javacardx.crypto.RandomData
Generates random data.
generateDigest(byte[], short, short, byte[], short). Method in class javacardx.crypto.MessageDigest
generates a hash of the input data.
generateDigest(byte[], short, short, byte[], short). Method in class javacardx.crypto.Sha1MessageDigest
generates a hash of the input data using the SHA1 algorithm.

generateMAC(byte[], short, short, byte[], short, byte). Method in class javacardx.crypto.DES3_Key
Generates a MAC using triple DES decryption in CBC mode.
generateMAC(byte[], short, short, byte[], short, byte). Method in class javacardx.crypto.DES_Key
Generates a MAC using single DES decryption in CBC mode.
generateMAC(byte[], short, short, byte[], short, byte). Method in class javacardx.crypto.SymKey
Generates a MAC using decryption in CBC mode.
getAID(). Static method in class javacard.framework.System
Returns the unique Applet Identifier (AID) object associated with the current applet execution context.
getAuthFlag(byte). Method in class javacardx.framework.FileSystem
Get authorization flag.
getBitLength(). Method in class javacardx.crypto.AsymKey
Gets the length of the key in bits.
getBlockSize(). Method in class javacardx.crypto.SymKey
Gets the block size used by the algorithm associated with this key.
getBuffer(). Method in class javacard.framework.APDU
Returns the APDU buffer byte array.
getChildFile(byte). Method in class javacardx.framework.DedicatedFile
Get the File object for the specified child file.
getCurrentDedicatedFile(). Method in class javacardx.framework.FileSystem
Get current DF.
getCurrentElementaryFile(). Method in class javacardx.framework.FileSystem
Get current EF.
getCurrentRecNum(). Method in class javacardx.framework.FileSystem
Get current record number.
getData(). Method in class javacardx.framework.TransparentFile
Gets the byte array containing the data for this file.
getData(APDU). Method in class javacardx.framework.FileSystem
Handles GET DATA command APDU as specified by ISO 7816-4.
getFCI(). Method in class javacardx.framework.File
Get this file’s FCI (if any).
getFID(). Method in class javacardx.framework.File
Get this file’s 16-bit FID.
getFileSystem(). Method in class javacardx.framework.File
Get the file system object (if any) which this file belongs to
getInBlockSize(). Static method in class javacard.framework.APDU
Returns the configured incoming block size.
getKeyLength(). Method in class javacardx.crypto.SymKey
Gets the length of the key.
getMaxChildFiles(). Method in class javacardx.framework.DedicatedFile
Get the maximum number of child files in this DF.
getMaxCommitCapacity(). Static method in class javacard.framework.System
Returns the total number of bytes in the commit buffer.
getMaxNumRecords(). Method in class javacardx.framework.LinearVariableFile
Get the maximum number of records in this file.
getNAD(). Method in class javacard.framework.APDU
Returns the T=1 transport protocol Node Address byte, NAD.T=0 returns 0.
getName(). Method in class javacardx.framework.DedicatedFile
Get the file’s name