

# JCOP30 Technical Brief



**Overview:** This document contains a simple overview about the technical capabilities of the first contact/contactless (dual-interface) member of the JCOP card OS family. Requests for further information may be directed at [javacard@zurich.ibm.com](mailto:javacard@zurich.ibm.com).

## 1. Basic specifications

JCOP is an IBM BlueZ implementation of the basic specifications [1] and [2] including refinements from Visa International set in the Visa OpenPlatform Card Implementation Guides (<http://www.visa.com/nt/suppliers/vendor>). All necessary clarifications from ISO7816 and EMV 2000 are also incorporated into the implementation where so required by [1] and [2].

JCOP30 is the first dual-interface member of this family. It conforms to the VOP Card Implementation Guide 2.1.3 Compact from August 2000.

Its successor, JCOP30v2.0 is compliant with the Visa OpenPlatform Card Implementation Requirements Configuration 2 with PK, Version 2.0 from February 2002.



## 2 Communications

### 2.1 Supported protocols

ISO7816 T=1 direct convention [default]<sup>1</sup>

ISO7816 T=0 direct convention<sup>1</sup>

ISO7816 T=1 inverse convention<sup>1</sup>

ISO7816 T=0 inverse convention<sup>1</sup>

ISO14443A T=CL

### 2.2 Supported speeds

#### 2.2.1 Contact protocols:

At the default clock rate of 3.57 MHz, the following communication speeds can be attained:

9600 bit/sec [default]

19200 bit/sec

38400 bit/sec

57600 bit/sec

115200 bit/sec

#### 2.2.2 Contactless protocol:

106000 bit/sec

212000 bit/sec

424000 bit/sec

848000 bit/sec<sup>2</sup>

---

<sup>1</sup> The contact protocols of JCOP can be configured to support the clock-stop feature of certain terminals to save power consumption (typically done in mobile phones). This feature is available since JCOP30v2.

<sup>2</sup> not approved for use in production cards

## 3 Memory availability for applications

### 3.1 EEPROM

#### 3.1.1 Persistent Java heap

Used for allocating persistent objects, applets, and storage of post-issuance loaded applet code (aka packages).

Size: 14kB (without custom ROM applets). Only for JCOP30v1: 13kB

#### 3.1.2 Transaction buffer

Used to save data written transactionally, e.g. all persistent byte and short stores, as well as persistent parameters to Util.arrayCopy; see [1].

Size: 512 bytes

### 3.2 RAM

#### 3.2.1 Transient Java heap

Used for allocating transient objects and arrays of type CLEAR\_ON\_RESET and CLEAR\_ON\_DESELECT.

Size: 487 bytes

#### 3.2.2 APDU buffer

Used to hold incoming and outgoing communications data.

Size: 261 bytes

#### 3.2.3 Java stack

Used to hold call parameters, local variables, and stack frames of the VM.

Size: 200 bytes

### 3.3 ROM

24kB free for applications\*

---

\* Custom applets may be submitted to Philips for inclusion into a ROM mask (see section 5)

## 4 Supported optional features

Certain features listed in [1] and [2] are not defined to be mandatory. The ones implemented in JCOP are listed below.

### 4.1 JavaCard

#### 4.1.1 Garbage Collection

Fully implemented: Deleted objects, applets, and packages are fully reclaimed and the space can be used for other purposes after deletion.

#### 4.1.2 Cryptographic Algorithms

JCOP30 has the ability to generate RSA keys on the card. The following JavaCard API constants (see [1]) are implemented by this version of JCOP:

##### Ciphers:

ALG\_DES\_CBC\_NOPAD  
ALG\_DES\_CBC\_ISO9797\_M1  
ALG\_DES\_CBC\_ISO9797\_M2  
ALG\_DES\_ECB\_NOPAD  
ALG\_DES\_ECB\_ISO9797\_M1  
ALG\_DES\_ECB\_ISO9797\_M2  
ALG\_RSA\_PKCS1  
ALG\_RSA\_NOPAD

##### Signatures:

ALG\_DES\_MAC8\_NOPAD  
ALG\_DES\_MAC8\_ISO9797\_M1  
ALG\_DES\_MAC8\_ISO9797\_M2  
ALG\_RSA\_SHA\_ISO9796  
ALG\_RSA\_SHA\_PKCS1  
ALG\_RSA\_MD5\_PKCS1

##### MessageDigest:

ALG\_SHA  
ALG\_MD5

**RandomData:**

ALG\_SECURE\_RANDOM

**KeyTypes:**

LENGTH\_DES

LENGTH\_DES3\_2KEY

LENGTH\_RSA\_2048<sup>1</sup> (JCOP30v2 only)LENGTH\_RSA\_1024<sup>1</sup>LENGTH\_RSA\_768<sup>1</sup>LENGTH\_RSA\_512<sup>1</sup>

TYPE\_DES\_TRANSIENT\_RESET

TYPE\_DES\_TRANSIENT\_DESELECT

TYPE\_DES

TYPE\_RSA\_PUBLIC

TYPE\_RSA\_PRIVATE<sup>2</sup>

TYPE\_RSA\_CRT\_PRIVATE

**KeyPair:**

ALG\_RSA\_CRT

**4.1.3 APDU class**

The method `APDU.getProtocol()` returns according to [1] the currently activated communications protocol. In compliance with [1], JCOP30 returns `APDU.PROTOCOL_T0` (0) if T=0 is running, and `APDU.PROTOCOL_T1` (1) if T=1 is running. JCOP30 returns none of these constants if T=CL is running. Hence, using a query of the form

```
if ((APDU.getProtocol() != APDU.PROTOCOL_T0) && (APDU.getProtocol() != APDU.PROTOCOL_T1))
```

can be used to cease computation in an applet that does not wish to execute if run over the contactless interface.

**4.2 OpenPlatform****4.2.1 Global PIN**

Fully implemented: All described APDU and API interfaces for this feature are present.

<sup>1</sup> All multiples of 32 (bit) are supported as valid RSA key lengths. Thus, key length values such as 736 (bits) can be passed as parameters to the respective functions.

<sup>2</sup> Private Keys must be loaded with key material. On-card key generation is only supported for RSA keys in CRT format

## 5 Supported Hardware

The supported configuration includes a 1kB Mifare Emulation mode (“Mifare Standard”) ensuring interoperability of JCOP30 in existing Mifare infrastructures. The usual JCOP ROM applet integration facilities can be used to create custom masks\*.

### 5.1 Philips P8RF5016

64 kB ROM → 24kB free for ROM'd applets in Custom Mask Process

16 kB EEPROM

2300 Bytes RAM

Triple-DES coprocessor

FameX RSA coprocessor

Mifare Standard Emulation

---

\* Custom applets may be submitted to Philips for inclusion into a ROM mask. The maximum package size is 16kB. The Custom Mask process is only available starting with JCOP30v2.

## 6 Performance figures

In the absence of standard performance tests, typical applet's operations are timed. The protocol used is T=1 at 9600 bit/sec. The reader clocks the chip at 3.71 MHz. The applets are the Visa approved versions after having been initialized and populated with keys as required for Visa VTF testing. To avoid measuring communications overhead, timing is measured between the last APDU byte sent to the reader and the first byte returned from the card.

Operation	ETUs	msec
SELECT CardManager*	82	7.6
INIT UPDATE CardManager	276	25.7
EXTERNAL AUTH CardManager	191	17.8
Install VisaCash	4198	390.4
SELECT VisaCash*	88	8.2
Initialize LOAD for VisaCash	416	38.7
Perform LOAD for VisaCash	895	83.2
Initialize DEBIT for VisaCash	116	10.8
Perform DEBIT for VisaCash	894	83.1
ReadBalance from VisaCash	52	4.8
SELECT VSDC*	182	16.9
GenerateAC from VSDC	1609	149.6

The PK operations are largely dominated by the hardware speed of the FameX. Note that on-card key generation is a random-based process; thus the figure given is only an average value. Values are measured at low-level; depending on Java programming skills, application-level code can add some time to the values depicted here.

Operation	Msec
Generate 1024bit CRT key	~ 5000
1024 bit CRT public key operation (F4)	32
1024 bit CRT private key operation	220

\* Not first SELECT to eliminate potential applet setup effects

## A Revision History

- 0.9 Preliminary version (prior to Visa approval)
- 0.91 AIDs or ROMed applets added
- 0.92 References to external specifications added
- 0.93 Incorrect reference to PseudoRandom constant deleted (4.1.2)
- 1.0 Document reformatted
- 1.1 Timing explanations added (6); ROM masking sentence added (5)
- 1.2 Information on supported RSA key lengths added (4.1.2)
- 1.3 Information on free ROM updated (3.3) and Information on APDU class and usage of T=CL added (4.1.3)
- 2.0 Specification update to cover JCOP30v2 (1, 4.1.2, 3.1.1); correction of RAM heap size (3.2.1, 1); caveat on T=CL speed added (2.2.2); added previously omitted KeyPair constant (4.1.2)
- 2.1 Mask/Free ROM applet size clarified (5)
- 2.2 Specification amendment stating explicitly that RSA private keys are supported (4.1.2)
- 2.3 Added comment on clock-stop feature (2.1)
- 2.4 Logo Update
- 2.5 Corrected free Heap size (3.2.1)

## **B**      **References**

- [1] Sun Microsystems: JavaCard 2.1.1 <http://java.sun.com/products/javacard>
- [2] Global Platform Consortium: OpenPlatform 2.0.1' <http://www.globalplatform.org/>