

Ciontek

# Cashless Development Specification for CM30

V2.0.2

Updated on  
2025/07/08

• 1. Introduction	4
• 2. Revisions	4
• 3. Import the Cashless SDK for Android studio	5
• 4. Cashless Setup	6
4.1. Setup Address	6
4.2. setPeripheralId	6
4.3. setConfiguration	7
4.4. setOptionalFeature	8
4.5. registerMonitor	9
• 5. Cashless Start & Destroy	9
• 6. Cashless Event Monitor	10
6.1. onInitialComplete	11
6.2. onReset	12
6.3. onSetupMaxMinPrices	12
6.4. onVendRequest	13
6.5. onVendCancel	14
6.6. onVendSuccess	15
6.7. onVendFailure	17
6.8. onSessionComplete	18
6.9. onCashSale	18
6.10. onNegativeVendRequest	21
6.11. onSelectionDenied	21
6.12. onCouponReply	22
6.13. onReaderDisable	23
6.14. onReaderEnable	23
6.15. onReaderCancel	23
6.16. onReaderDataEntryResponse	23
6.17. onRevalueRequest	24
6.18. onRevalueLimitRequest	25
6.19. onSyncTimeDate	25
6.20. onDiagnostics	26
• 7. Cashless APIs	26
7.1. sendACK	26

7.2. sendJustReset	26
7.3. sendTimeDateRequest	27
7.4. sendDisplayRequest	27
7.5. sendBeginSession	28
7.6. sendSessionCancelRequest	30
7.7. sendVendApproved	31
7.8. sendVendDenied	32
7.9. sendMalfunctionion	32
7.10. sendRevalueApproved	33
7.11. sendRevalueDenied	33
7.12. sendRevalueLimitAmount	34
7.13. sendDataEntryRequest	35
7.14. sendDataEntryCancel	36
7.15. sendSelectionRequest	36
7.16. sendCouponReport	38
7.17. sendDiagnosticsResponse	38
• 8. Return Code	39
• 9. Appendix	39
9.1. Appendix1 Reset Sequence	39
9.2. Appendix2 Vend Sequence	40
9.3. Appendix3 Negative Vend Sequence	42

- **1. Introduction**

CM30 is a cashless device connect to VMC by MDB Slave port that meets Multi-Drop Bus/Internal Communication Protocol .This document is the instruction of all APIs defined by Ciontek for developers to program their own Cashless application upon CM30.

Base on <MDB/ICP Version 4.3, Part Cashless Device>

**CM30 Cashless Address: 10H or 60H**

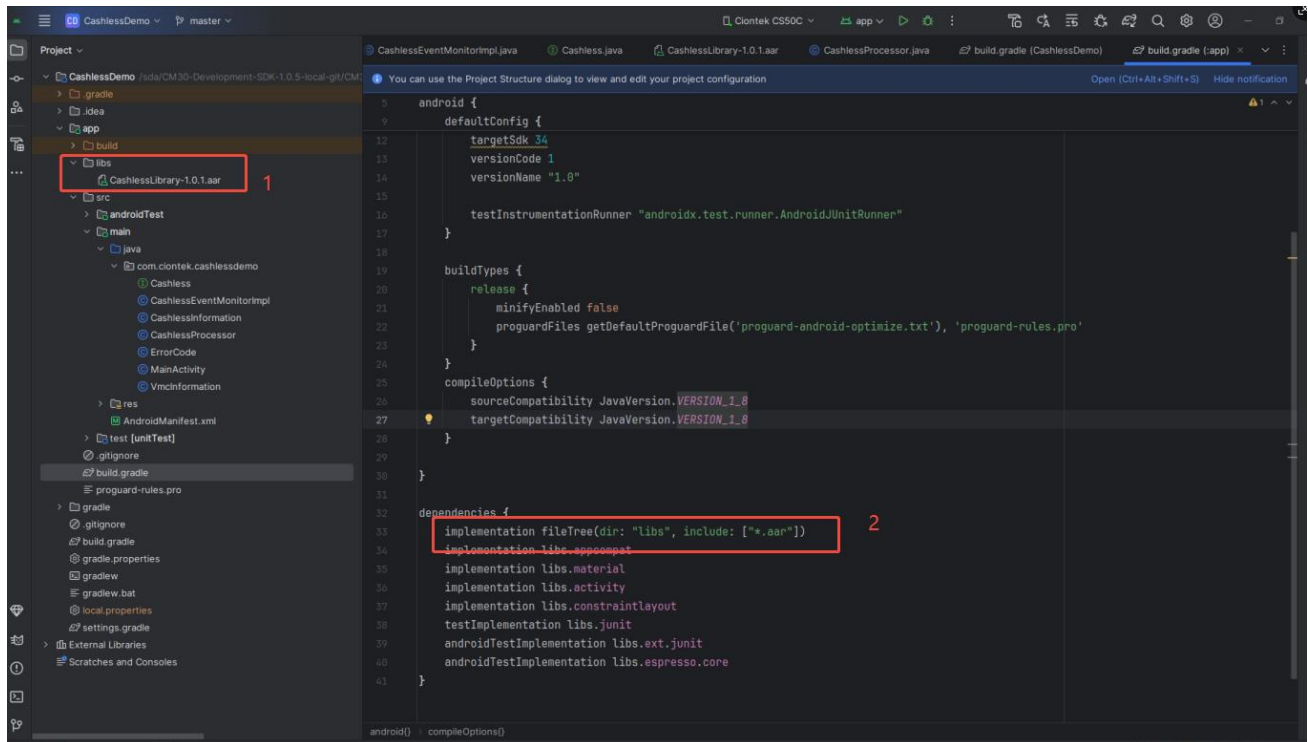
**NOTE:** The Android firmware no earlier than a62cu\_v0.13\_20250108g

- **2. Revisions**

version	author	date	remarks
V2.0.1	Tao	2025-01-06	
V2.0.2	Tao	2025-07-08	Add address api

### • 3. Import the Cashless SDK for Android studio

Import the “CashlessLibrary-2.0.1.aar” to your project, as shown in the picture below:



#### Call example:

‘CashlessManager’ describes all the APIs for Cashless device.

Get CashlessManager instance:

```
private CashlessManager mCashlessManager = CashlessManager.getInstance();
```

‘ICashlessEventMonitor’ listening all the events that comes from Vending Machine Controller.

**For more details please refer to the demo ‘CashlessDemo.zip’.**

## • 4. Cashless Setup

After get a cashless instance, you need to setup cashless by your actual application scenario.

For example, to setup the cashless's Feature Level/Currency Code/Scale Factor/Decimal Places and some option features and so on.

**NOTE:** Setup APIs should be called before start()

### 4.1. Setup Address

Setup the cashless address to 0x10 or 0x60. If ignore, the default 0x10 would be used.

**NOTE:** should be called before start()

Setup the cashless address

setAddress	int address
	0x10 or 0x60
return	0 -> Success <0 -> Fail

Get the cashless address

getAddress	none
return	0x10 or 0x60

Example:

```
/**
 * set cashless address to 0x60
 */
mCashlessManager.setAddress(0x60);

/**
 * get cashless address
 */
int address = mCashlessManager.getAddress();
```

### 4.2. setPeripheralId

Setup the cashless peripheral ID information, like Manufacturer Code/Serial Number/Model Number.

If ignore, the default would be used.

**NOTE:** should be called before start()

setPeripheralId	byte[] manufacturerCode	byte[] serialNumber	byte[] modelNumber
byte count	3 byte	12 bytes	12 bytes
return	0 -> Success <0 -> Fail		

#### **Manufacturer Code** - 3 byte ASCII

Identification code for the equipment supplier. Currently defined codes are listed in the EVA document entitled "European Vending Association Data Transfer Standard" (EVA-DTS), the Audit Data Lists section, sub-section 2, "Manufacturer Codes".

#### **Serial Number** - 12 bytes ASCII

Factory assigned serial number.

#### **Model Number** - 12 bytes ASCII

Manufacturer assigned model number.

Example:

```
byte[] manufacturerCode = {'c','t','k'};
byte[] serialNumber = {'c','t','k','-','1','2','3','4','5','6','7','8'};
byte[] modelNumber = {'c','m','3','0','1','2','3','4','5','6','7','8'};
mCashlessManager.setPeripheralId(manufacturerCode,serialNumber,modelNumber);
```

## 4.3. setConfiguration

Setup the cashless FeatureLevel/Currency Code/Scale Factor/Decimal Places/Max Non Response Time/Misc Options.

**NOTE:** should be called before start()

setConfiguration	int featureLevel	int currencyCode	int scaleFactor	int decimalPlaces	int MaxNonRespTime	int misc
byte count	2 bytes	1 byte	1 byte	1 byte	1 byte	1 byte
return	0 -> Success <0 -> Fail					

#### **Feature Level:**

Indicates the feature level of the cashless. Currently feature levels are:

01/02/03

#### **Country / Currency Code: 2 byte**

The packed BCD country / currency code of the cashless can be sent in two different forms depending on the value of the left most BCD digit. If the left most digit is a 0, the International Telephone Code is used to indicate the country that the reader is set-up for. For example, the USA code is 00 01H (high byte = 00 and low byte = 01). If the left most digit is a 1, the latest version of the ISO 4217 numeric currency code is used. For example, the code for the US dollar is 18 40H (high byte = 18 and low byte = 40) and for the Euro is 19 78 (high byte = 19 and low byte = 78). Use FFFFh if the country code is unknown.

#### **Scale Factor: 1 byte**

The multiplier used to scale all monetary values transferred between the VMC and the cashless.

#### Decimal Places: 1 byte

The number of decimal places used to communicate monetary values between the VMC and the cashless. All pricing information sent between the VMC and the payment media reader(cashless) is scaled using the scale factor and decimal places. This corresponds to:

$$\text{ActualPrice} = P * X * 10^{-Y}$$

where P is the scaled value send in the price bytes, and X is the scale factor, and Y is the number of decimal places. For example if there are 2 decimal places and the scale factor is 5, then a scaled price of 7 will mean an actual of 0.35.

#### MaxNonRespTime: 1byte

Application maximum non response timeout. Default is 5 seconds. If time out the VMC will issue {@onReset} to cashless.

#### Miscellaneous Options: 1 byte - xxxxyyyy

xxxx: Unused (must be set to 0)

yyyy: Option bits

b0=0: NOT support of restoring funds to the user' s payment media or account. Do not request refunds.

b0=1: support of restoring funds to the user' s payment media or account. Refunds may be requested.

b1=0: NOT support multivend .Terminate session after each vend.

b1=1: support multivend. Multiple items may be purchased within a single session.

b2=0: Cashless devcie does NOT have a display.

b2=1: Cashless device does have its own display.

b3=0: does NOT support the VEND/CASH SALE .

b3=1: support the VEND/CASHSALE .

b4-b7=0 : Reserved

Example:

```
/**
 * config cashless feature,do it before call{@start}
 * featureLevel = 0x03 //cashless level3 feature
 * currencyCode = 0x1840 //US dollar
 * scaleFactor = 0x01 //scale factor is 1
 * decimalPlaces = 0x01 // there are 2 decimal places
 * MaxNonRespTime = 0x05; //max non response timeout is 5s
 * misc = 0x0E; // support MultiVend and CashSale
 */
mCashlessManager.setConfiguration(0x03,0x1840,0x01,0x02,0x05,0x0E);
```

## 4.4. setOptionalFeature

Set Level 03 option features can be support. If ignore, default 0x0000 is used.

The features described here requires the cashless device and the VMC both are level 03.

**NOTE:** should be called before start()



setOptionalFeature	int option
byte count	4 byte
return	0 -> Success <0 -> Fail

### Level 03 Readers

#### Optional Feature Bits - 4 bytes

b0 - File Transport Layer supported

b1 - 0 = 16 bit monetary format, 1 = 32 bit monetary format

b2 - support multi currency / multi lingual

b3 - support Negative Vend

b4 - support data entry

b5 - support “Always Idle”

b6 - support “Remote Vend” to initiate dispensing process without the vending machine user interface .

b7 - support “Basket / Partial Refund / Options Price ” feature to perform multiple vends with single VEND REQUEST and multiple VEND SUCCESS / VEND FAILURES.

b8 - allow “Coupon” feature

b9 - allow “Ask Begin Session” feature in selection first (b5 should also set to 1)

b10 - allow “Enhanced Item Number Information”. add Item Number Dispensed and the EVA-DTS PA101 fields for item Selected and Dispensed to the Vend Success event and Cash Sale event.

b11 to b31 not used (should be set to 0)

Example:

```
// ' Always Idle' b5 set 1
CashlessManager.setOptionalFeature(0x20);
```

## 4.5. registerMonitor

Register an monitor for listening all the events issued by the Vending Machine Controller.

Example:

```
import android.hardware.cashless.ICashlessEventMonitor;
mCashlessManager.registerMonitor(mCashlessEventMonitorImpl);
```

Developers can extends the calss ‘ICashlessEventMonitor’ and implement their own event monitor in their application. (See Section 6 for more details)

## • 5. Cashless Start & Destroy

After Setup , Call the API {@Start} to run cashless device, And call {@Destroy} to stop the cashless when exit your application.

If {@ Start} called, the API {@ isActive} will return true;and {@ Destroy } called, {@ isActive} return false.

Example:

```
/**
 * start cashless when your app start
 */
mCashlessManager.start();

/**
 * stop cashless when your app exit
 */
mCashlessManager.destroy();

/**
 * get state
 */
mCashlessManager.isActive();
```

## • 6. Cashless Event Monitor

'ICashlessEventMonitor' describes all the events that comes from Vending Machine Controller.

```
interface ICashlessEventMonitor {
    void onInitialComplete(byte[] cashlessInfo, byte[] vmcInfo);
    void onReset();
    void onSetupMaxMinPrices(byte[] data);
    void onVendRequest(byte[] data);
    void onVendCancel();
    void onVendSuccess(byte[] data);
    void onVendFailure(byte[] data);
    void onSessionComplete();
    void onCashSale(byte[] data);
    void onNegativeVendRequest(byte[] data);
    void onSelectionDenied(byte[] data);
    void onCouponReply(byte[] data);
    void onReaderDisable();
    void onReaderEnable();
    void onReaderCancel();
    void onReaderDataEntryResponse(byte[] dataEntry);
    void onRevalueRequest(byte[] data);
    void onRevalueLimitRequest();
    void onSyncTimeDate(byte[] timeDate);
    void onDiagnostics(byte[] data);
```

```
}
```

## 6.1. onInitialComplete

Indicates the cashless device initial complete. And reported the cashless device information and the VMC information.

**Reply with API:** {@ none}

onInitialComplete	byte[] cashlessInfo	byte[] vmcInfo
	Y0-Y50	Y0-Y32
Reply with API	{@ none}	

### Cashless device Information - 51 byte

Y0: Cashless Feature Level

Y1-Y2: Currency Code

Y3: Scale Factor

Y4: Decimal Places

Y5: MaxNonResponseTime

Y6: Misc Options

Y7-Y8: Max Price

Y9-Y10:MinPrice

Y11: State of the feature 'File Transport Layer' , =1 means enable ; =0 means disable.

Y12: =0 means 16 bit monetary format, =1 means 32 bit monetary format

Y13: State of the feature 'multi currency / multi lingual', =1 means enable ; =0 means disable.

Y14: State of the feature 'Negative Vend', =1 means enable ; =0 means disable.

Y15: State of the feature 'Data Entry', =1 means enable ; =0 means disable.

Y16: State of the feature 'Always Idle', =1 means enable ; =0 means disable.

Y17: State of the feature 'Remote Vend', =1 means enable ; =0 means disable.

Y18: State of the feature 'Basket / Partial Refund / Options Price',=1 means enable ; =0 means disable.

Y19: State of the feature 'Coupon', =1 means enable ; =0 means disable.

Y20: State of the feature 'Ask Begin Session',=1 means enable ; =0 means disable.

Y21: State of the feature 'Enhanced Item Number Information', =1 means enable ; =0 means disable.

Y22-Y24: Manufacturer Code

Y25-Y36: Serial Number

Y37-Y48: Model Number

Y49-Y50: Software Version

### VMC information - 33 bytes

Y0: VMC Feature Level

Y1: Columns on Display.The number of columns on the display. Set to 00H if the display is not available to the reader.

Y2: Rows on Display. The number of rows on the display

Y3: Display Information - xxxxyyyy

xxxxx = Unused

yyy = Display type

000 : Numbers, upper case letters, blank and decimal point.

001 : Full ASCII

010-111: Unassigned

Y4-Y6: Manufacturer Code of VMC

Y7-Y18: Serial Number of VMC

Y19-Y30: Model Number of VMC

Y31-Y32: Software Version of VMC

## 6.2. onReset

If this event is received by a cashless device it should terminate any ongoing transaction (with an appropriate credit adjustment, if appropriate), eject the payment media (if applicable).

**Reply with API** {@ sendJustReset}

onReset	none
Reply with API	{@ sendJustReset}

**Level 01/02/03 Readers**

## 6.3. onSetupMaxMinPrices

Indicates the VMC is sending the price range to the cashless.

**Reply with API** {@ none}

onSetupMaxMinPrices	byte[] data	
	Maximum Price Y0-Y1	Minimum Price Y2-Y3
Reply with API	{@ none}	

**Level 01/02/03 Readers**

Y0-Y1: Maximum Price - scaled

This information should be sent as soon as the VMC prices have been established and any time there is a change in the maximum price, If the VMC does not know the maximum price, FFFFh should be sent.

Y2-Y3: Minimum Price - scaled

This information should be sent as soon as the VMC prices have been established and any time there is a change in the minimum price. If the VMC does not know the minimum price, 0000h should be sent.

If "Expanded Currency Mode" enabled.

onSetupMaxMinPrices	byte[] data
---------------------	-------------

	Maximum Price Y0-Y3	Minimum Price Y4-Y7	Currency Code Y8-Y9
Reply with API	{@ none}		

### Level 03 (EXPANDED CURRENCY MODE enabled) Readers

Y0-Y3: Maximum Price - scaled

Same with above

Y4-Y7: Minimum Price - scaled

Same with above

Y8-Y9: Currency Code

The currency code used during this command per ISO 4217. The value is configured as packed BCD with the leading digit a 1 (one). For example, the code for the US dollar would be 1840 (Y8 = 18 and Y9 = 40). and for the Euro is 1978 (Y8 = 19 and Y9 = 78).

NOTE: If 32 bit monetary format and or multi currency / multi lingual (b2) options are enabled, this condition will be known as **EXPANDED CURRENCY MODE** in the rest of the document.

## 6.4. onVendRequest

The patron has made a selection. The VMC is requesting vend approval from the payment media reader before dispensing the product.

**Reply with API:** {@sendVendApproved} or {@ sendVendDenied }

onVendRequest	byte[] data	
	Item Price Y0-Y1	Item Number Y2-Y3
Reply with API	{@sendVendApproved} or {@ sendVendDenied }	

### Level 01/02/03 Readers

Y0-Y1 : Item Price - scaled

The price of the selected product.

Y2-Y3 : Item Number

The item number of the selected product. This number is defined by the manufacturer, and set to FFFFh for undefined or not implemented.

If "EXPANDED CURRENCY MODE" enabled

onVendRequest	byte[] data	
	Item Price Y0-Y3	Item Number Y4-Y5
Reply with API	{@sendVendApproved} or {@ sendVendDenied }	

### Level 03 (EXPANDED CURRENCY MODE) Readers

Y0-Y3 : Item Price - scaled

The price of the selected product.

Y4-Y5 : Item Number

The item number of the selected product. This number is defined by the manufacturer, and set to FFFFh for undefined or not implemented.

onVendRequest	byte[] data				
	Item Price Y0-Y1	ItemNumber Y2-Y3	Unused Y4-Y7	Item Count Y8	Options Price Y9-Y10
	(Y0-Y3 @32bit)	(Y4-Y5 @32bit)	(Y6-Y9 @32bit)	(Y10 @32bit)	(Y11-Y14 @32bit)
Reply with API	{@sendVendApproved} or {@ sendVendDenied }				

**Level 03 ("basket / partial refund / options price" enabled) readers**

**Remark: "@32bit" means EXPANDED CURRENCY MODE enabled.**

Y0-Y1 : Item Price - scaled

(Y0-Y3 @32bit) The price of the selected product or the amount for this vend.

Y2-Y3 : Item Number

(Y4-Y5 @32bit): The item number of the selected product. This number is defined by the manufacturer, and set to FFFFh for undefined or not implemented. Vend request which is initiated by SELECTION REQUEST response should have the same ITEM NUMBER as ITEM NUMBER in the SELECTION REQUEST response.

Y4-Y7 : Bytes intentionally skipped/excluded - can be set to 00h

(Y6-Y9 @32bit) Bytes intentionally skipped/excluded - can be set to 00h

Y8 : Item Count

(Y10 @32bit) Number of items to be dispensed. Should be set to 01H for normal vend(only one product will be dispensed and only one VEND SUCCESS or VEND FAILURE will be issued) and 02H or above for shopping basket vend mode (when a number of products will be dispensed and VMC will issue a VEND SUCCESS or VEND FAILURE for each product). For shopping basket vend mode, item number should be set to FFFFh and item price should be a sum of all products to be dispensed.

Y9-Y10 : Options price

(Y11-Y14 @32bit) The total price of included options (extra sugar, cup and so on) - scaled. Part of item price that covers paid options. This price is already included to field "item price" and just informs cashless device about options cost (for case when "price holding in cashless device" mode is used). For example, if product price (scaled) without options is 5 and with options is 6, the field "Item price" should be 6 and "Options price" should be 1.

## 6.5. onVendCancel

This event indicates the {@ onVendRequest} would be cancel before a {@ sendVendApproved}/{@ sendVendDenied} has been sent by cashless device.

**Reply with API:** {@ sendVendDenied }

onVendCancel	none
Reply with API	{@ sendVendDenied }

**Level 01/02/03 Readers**

## 6.6. onVendSuccess

This event indicates the selected product has been successfully dispensed.

**Reply with API:** {@ none}

onVendSuccess	byte[] data
	Item number Y0-Y1
Reply with API	{@ none}

**Level 01/02/03 Readers**

Y0-Y1 : Item number

The item number of the selected product. This number is defined by the manufacturer, and set to FFFFh for undefined or not implemented.

If “Enhanced Item Number Info” enabled

onVendSuccess	byte[] data			
	Item number Selected Y0-Y1	Item number Dispensed Y2-Y3	PA101 Item Selected Y4-Y9	PA101 Item Dispensed Y10-Y15
Reply with API	{@ none}			

**Level 03 (“Enhanced Item Number Info” enabled) readers**

If “basket / partial refund / options price” enabled

onVendSuccess	byte[] data			
	Item number Y0-Y1	Vend Amount Y2-Y3 (Y2-Y5 @32bit)	Item Count Y4 (Y6 @32bit)	Options Amount Y5-Y6 (Y7-Y10 @32bit)
Reply with API	{@ none}			

**Level 03 (“basket / partial refund / options price” enabled) readers**

If “Enhanced Item Number Info ” and “basket / partial refund / options price” both enabled

onVendSuccess	byte[] data						
	Item number Selected	Item number Dispensed	PA101 Item Selected	PA101 Item Dispensed	Vend Amount Y16-Y17	Item Count Y18	Options Amount Y19-Y20

	Y0-Y1	Y2-Y3	Y4-Y9	Y10-Y15	(Y16-Y19 @32bit	(Y20 @32bit)	(Y21-Y24 @32bit)
Reply with API	{@ none}						

**Level 03 (“Enhanced Item Number Info” & “basket / partial refund / options price” enabled) readers**

**Remark: “@32bit” means EXPANDED CURRENCY MODE enabled.**

#### **Y0-Y1: Item Number**

if no Enhanced Item Number Info enabled, As a “best practice” the Item Number should be the dispensed product. This number is defined by the manufacturer, and set to FFFFh for undefined or not implemented. This number could be different from the item number in the corresponding VEND REQUEST. For example, if the product selected is sold out and the machine knows that the same product is in a different location, the actual vended location should be sent. For basket vend mode VEND SUCCESS (or VEND FAILURE) is issued for each product in the basket. In this case this field also indicates actual dispensed selection for each product.

#### **Y0-Y1: Item Number Selected**

The item number of the selected product. This number is defined by the manufacturer, and set to FFFFh for undefined or not implemented. The number refers to the physical button or button sequence used to make a selection. For basket vend mode VEND SUCCESS (or VEND FAILURE) is issued for each product in the basket. In this case this field also indicates actual selection for each product.

#### **Y2-Y3: Item Number Dispensed**

The item number of the dispensed product. This number is defined by the manufacturer, and set to FFFFh for undefined or not implemented. The number refers to the vend mechanism or physical vend location in the machine.

#### **Y4-Y9: EVA DTS PA101 Selected Item Number**

This is the same information that is sent in the PA101 field for the item selected. The number refers to the physical button or button sequence used to make a selection. Since the PA101 field can be 1-6 characters long and the MDB fields are fixed length, unused characters should be sent as trailing ASCII spaces (20h). All other characters are ASCII alpha/numeric.

#### **Y10-Y15: EVA DTS PA101 Dispensed Item Number**

This is the same information that is sent in the PA101 field for the item dispensed. The number refers to the vend mechanism or physical vend location in the machine. Since the PA101 field can be 1-6 characters long and the MDB fields are fixed length, unused characters should be sent as trailing ASCII spaces (20h). All other characters are ASCII alpha/numeric.

**Y2-Y3(or Y2-Y5 or Y16-Y17 or Y16-Y19) : Vend Amount – scaled.** The vend amount of the dispensed product. For normal mode it should be less (in case of partial dispensing) or equal to VEND AMOUNT of VEND APPROVED response. For basket mode, the sum of VEND AMOUNT of all VEND SUCCESS and VEND FAILURE requests should be less (in case if some products were dispensed partially) or equal to VEND AMOUNT of VEND APPROVED response.

**Y4(or Y6 or Y18 or Y20) :Item Count.** Number of products waiting for dispense in this vend if vend is a vend basket. For normal vend and for last product of basket vend this field should be 0. For basket mode this field should be decremented after each VEND SUCCESS / FAILURE. This field could be also 0 for not last product in basket mode. This means that basket vend was interrupted in the middle and no further VEND SUCCESS / FAILURE will be issued (the cashless device should



consider that all rest products was not dispensed).

**Y5-Y6(or Y7-Y10 or Y19-Y20 or Y21-Y24): Options Amount** - scaled (part of vend amount which covers paid options).The amount of the dispensed options. For normal mode it should be less (in case of partial dispensing) or equal to OPTIONS AMOUNT of VEND APPROVED response.For basket mode, the sum of OPTIONS AMOUNT of all VEND SUCCESS and VEND FAILURE requests should be less (in case if some products were dispensed partially) or equal to OPTIONS AMOUNT of VEND APPROVED response.

## 6.7. onVendFailure

A vend has been attempted at the VMC but a problem has been detected and the vend has failed. The product was not dispensed. Funds should be refunded to user's account.

And in order to ensure that a reader refunds after a Vend Failure event, if refunds success,cashless reply with {@sendACK}, if refunds failure cashless reply with {@sendMalfunctiontion}.

**Reply with API:** {@ sendACK} or {@ sendMalfunctiontion}

onVendFailure	byte[] data
	null
Reply with API	{@ sendACK} or {@ sendMalfunctiontion}

### Level 01/02/03 Readers

If “basket / partial refund / options price” enabled.

onVendFailure	byte[] data				
	Item number Y0-Y1	Vend Amount Y2-Y3 (Y2-Y5 @32bit)	Item Count Y4 (Y6 @32bit)	Options Amount Y5-Y6 (Y7-Y10@32bit)	Reason Y7 (Y11 @32bit)
Reply with API	{@ sendACK} or {@ sendMalfunctiontion}				

### Level 03 (“basket / partial refund / options price” enabled) readers

**Remark: “@32bit” means EXPANDED CURRENCY MODE enabled.**

#### Y0-Y1: Item Number

The item number of the failed product. This number is defined by the manufacturer, and set to FFFFh for undefined or not implemented.

#### Y2-Y3 (or Y2-Y5) : Vend amount-scaled

Vend amount (scaled) of the product failed to be dispensed (in terms of VEND AMOUNT of VEND APPROVED). For shopping basket vend mode,this value should be accumulated by the cashless device for all products with VEND FAILURE for case of unpredictable reset from VMC before the last product will be confirmed by VMC. In the case of unpredictable RESET, the cashless device should consider that all not confirmed products was successfully dispensed, and thus it should refund back only value accumulated with all VEND FAILURE requests.

**Y4(or Y6): Item Count**

Number of products waiting for dispense in this vend if vend is a vend basket. For normal vend and for last product of basket vend this field should be 0. For basket mode this field should be decremented after each VEND SUCCESS / FAILURE. This field could be also 0 for not last product in basket mode. This means that basket vend was interrupted in the middle and no further VEND SUCCESS / FAILURE will be issued (the cashless device should consider that all rest products was not dispensed).

**Y5-Y6(or Y7-Y10): Options amount-scaled**

Options amount (scaled) of the product failed to be dispensed (in terms of OPTIONS AMOUNT of VEND APPROVED) – part of vend amount which covers options. For shopping basket vend mode in price holding mode, this value should be accumulated by the cashless device for all products with VEND FAILURE for case of unpredictable reset from VMC before the last product will be confirmed by VMC. In the case of unpredictable RESET, the cashless device should consider that all not confirmed products was successfully dispensed, and thus it should refund back only value accumulated with all VEND FAILURE requests.

**Y7(or Y11):Reason**

00h: Unknown error / out of order.  
 01h: Selection doesn't exist / not configured.  
 02h: Selection empty / product was not dispensed.  
 03h: Selection defective.  
 04h: Ingredient is over  
 05h: Selection is blocked.  
 06h: Selection is inhibited (for this time or this pricelist or this user).  
 07h: Product is expired.  
 08h: Temperature conditions is out of range  
 09h: Machine is busy or is in the wrong mode (menu for example).  
 0Ah: insufficient credit

**6.8. onSessionComplete**

This tells the cashless that the session is complete.

**Reply with API:** {@ none}

onSessionComplete	none
Reply with API	{@ none}

**Level 01/02/03 Readers**

**6.9. onCashSale**

A cash sale (cash only or cash and cashless) has been successfully completed by the VMC.

**Reply with API:** {@ none}

onCashSale	byte[] data	
	Item Price Y0-Y1	Item Number Y2-Y3
Reply with API	{@ none}	

#### **Level 01/02/03 Readers**

If "Enhanced Item Number Info" enabled

onCashSale	byte[] data					
	Item Price Y0-Y1	Item Number Selected Y2-Y3	Item Number Dispensed Y4-Y5	PA101 Item Selected Y6-11	PA101 Item Dispensed Y12-Y17	Mixed Vend Flags Y18
Reply with API	{@ none}					

#### **Level 03 (Enhanced Item Number Info enabled) readers**

Y0-Y1: Item Price – scaled

The price of the selected product or cash portion of the price.

Y2-Y3: Item Number Selected

The item number of the selected product. This number is defined by the manufacturer, and set to FFFFh for undefined or not implemented. The number refers to the physical button or button sequence used to make a selection.

Y4-Y5: Item Number Dispensed

The item number of the dispensed product. This number is defined by the manufacturer, and set to FFFFh for undefined or not implemented. The number refers to the vend mechanism or physical vend location in the machine.

Y6-Y11: EVA DTS PA101 - Item Selected

This is the same information that is sent in the PA101 field for the item selected. The number refers to the physical button or button sequence used to make a selection. Since the PA101 field can be 1-6 characters long and the MDB fields are fixed length, unused characters should be sent as trailing ASCII spaces (20h). All other characters are ASCII alpha/numeric.

Y12-Y17: EVA DTS PA101 - Item Dispensed

This is the same information that is sent in the PA101 field for the item dispensed. The number refers to the vend mechanism or physical vend location in the machine. Since the PA101 field can be 1-6 characters long and the MDB fields are fixed length, unused characters should be sent as trailing ASCII spaces (20h). All other characters are ASCII alpha/numeric.

Y18: Mixed Vend Flags

b0: set if the item was (partially) paid with cash

b1: set if the item was (partially) paid with Cashless #1

b2: set if the item was (partially) paid with Cashless #2

b3-b7: reserved and should be 0

Note: b2-b0 bits should never be 000 or 010 for command sent to Cashless #1 and should never be 000 or 100 for command sent to Cashless #2.

If “Expanded Currency Mode” enabled

onCashSale	byte[] data		
	Item Price Y0-Y3	Item Number Y4-Y5	Item Currency Y6-Y7
Reply with API	{@ none}		

### Level 03 (Expanded Currency Mode) Readers

If “Expanded Currency Mode” and “Enhanced Item Number Info” enabled

onCashSale	byte[] data						
	Item Price Y0-Y3	Item Number Selected Y4-Y5	Item Currency Y6-Y7	Item Number Dispensed Y8-Y9	PA101 Item Selected Y10-Y15	PA101 Item Dispensed Y16-Y21	Mixed Vend Flags Y22
Reply with API	{@ none}						

### Level 03 (Expanded Currency Mode and Enhanced Item Number Info enabled) Readers

Y0-Y3: Item Price – scaled

The price of the selected product or cash portion of the price.

Y4-Y5: Item Number Selected

The item number of the selected product. This number is defined by the manufacturer, and set to FFFFh for undefined or not implemented. The number refers to the physical button or button sequence used to make a selection.

Y6-Y7: Item Currency

The currency for the item price used during the vend. This value may be converted within the reader to the readers balancing currency. The item currency is sent using the numeric code as defined in ISO 4217. The value is configured as packed BCD with the leading digit a 1 (one). For example, the code for the US dollar would be 1840 (Z6 = 18 and Z7 = 40). and for the Euro is 1978 (Z6 = 19 and Z7 = 78).

Y8-Y9: Item Number Dispensed

The item number of the dispensed product. This number is defined by the manufacturer, and set to FFFFh for undefined or not implemented. The number refers to the vend mechanism or physical vend location in the machine.

Y10-Y15: EVA DTS PA101 - Item Selected

This is the same information that is sent in the PA101 field for the item selected. The number refers to the physical button or button sequence used to make a selection. Since the PA101 field can be 1-6 characters long and the MDB fields are fixed length, unused characters should be sent as trailing ASCII spaces (20h). All other characters are ASCII alpha/numeric.

Y16-Y21: EVA DTS PA101 - Item Dispensed

This is the same information that is sent in the PA101 field for the item dispensed. The number refers to the vend mechanism or physical vend location in the machine. Since the PA101 field can be 1-6 characters long and the MDB fields are fixed length, unused characters should be sent as trailing ASCII spaces (20h). All other characters are ASCII alpha/numeric.

Y22: Mixed Vend Flags

b0: set if the item was (partially) paid with cash

b1: set if the item was (partially) paid with Cashless #1

b2: set if the item was (partially) paid with Cashless #2

b3-b7: reserved and should be 0

Note: b2-b0 bits should never be 000 or 010 for command sent to Cashless #1 and should never be 000 or 100 for command sent to Cashless #2.

## 6.10. onNegativeVendRequest

The patron has inserted an item. The VMC is requesting negative vend approval from the payment media reader before accepting the returned product.

**Reply with API:** {@ sendVendApproved} or {@ sendVendDenied}

onNegativeVendRequest	byte[] data	
	Item Value Y0-Y1	Item Number Y2-Y3
Reply with API	{@ sendVendApproved} or {@ sendVendDenied}	

### Level 03 (Negative Vend enabled) Readers

If "EXPANDED CURRENCY MODE" enabled

onNegativeVendRequest	byte[] data	
	Item Value Y0-Y3	Item Number Y4-Y5
Reply with API	{@ sendVendApproved} or {@ sendVendDenied}	

### Level 03 (EXPANDED CURRENCY MODE) Readers

#### Y2-Y3 : Item value – scaled

The value of the inserted product (16 Bit).

#### Y4-Y5 : Item Number

The item number of the inserted product. This number is defined by the manufacturer, and set to FFFFh for undefined or not implemented.

## 6.11. onSelectionDenied

A vend which was requested by SELECTION REQUEST could not be dispensed. The cashless device should keep it's state as it was before SELECTION REQUEST.

**Reply with API:** {@ none}

onSelectionDenied	byte[] data	
	Item Number Y0-Y1	Reason Y2

Reply with API	{@ none}
----------------	----------

### Level 03 (Remote Vend enabled) Readers

Y0-Y1 : Item number.

Should be equal to ITEM NUMBER of SELECTION REQUEST.

Y2 : Reason why it could not be dispensed.

00h: Unknown error / out of order.

01h: Selection doesn't exist / not configured.

02h: Selection empty / product was not dispensed.

03h: Selection defective.

04h: Ingredient is over

05h: Selection is blocked.

06h: Selection is inhibited (for this time or this pricelist or this user).

07h: Product is expired.

08h: Temperature conditions is out of range

09h: Machine is busy or is in the wrong mode (menu for example).

0Ah: insufficient credit

FFh: Machine is temporary busy (for example, waits for consumer when he will take his product or machine moves coffee brewer to initial position or it cleans itself and so on). The cashless device can repeat SELECTION REQUEST again within 1-2 seconds.

## 6.12. onCouponReply

Sent by VMC after "Coupon accepted" response after product dispensing (or after rejecting coupon). Should be sent before CASH SALE if used.

**Reply with API:** {@ none}

onCouponReply	byte[] data		
	Usage	Item Number	Used Value
	Y0	Y1-Y2	Y3-Y4 (Y3-Y6@32bit)
Reply with API	{@ none}		

### Level 03 (Coupon feature enabled) Readers

**Remark: "@32bit" means EXPANDED CURRENCY MODE enabled.**

Y0: Usage.

00 = coupon was rejected. 01 = product was dispensed.

Y1-Y2: Dispensed item number.

Y3-Y4(or Y3-Y6): Value (scaled) used to pay a dispensed product.

## 6.13. onReaderDisable

This informs the cashless that it has been disabled, i.e. it should no longer accept a patron's payment media for the purpose of vending. Vending activities may be re-enabled using the {@ onReaderEnable} event arrived. The cashless should retain all SETUP information.

NOTE: Any transaction in progress will not be affected and should continue to its normal completion.

**Reply with API:** {@ none}

onReaderDisable	none
Reply with API	{@ none}

**Level 01/02/03 Readers**

## 6.14. onReaderEnable

This informs the cashless that it has been enabled, i.e. it should now accept a patron's payment media for vending purposes.

**Reply with API:** {@ none}

onReaderEnable	none
Reply with API	{@ none}

**Level 01/02/03 Readers**

## 6.15. onReaderCancel

This event is issued to abort cashless activities which occur in the Enabled state.

**Reply with API:** {@ none}

onReaderCancel	none
Reply with API	{@ none}

**Level 01/02/03 Readers**

## 6.16. onReaderDataEntryResponse

The VMC is providing a DATA ENTRY RESPONSE to the cashless.

**Reply with API:** {@ none}

onReaderDataEntryResponse	byte[] dataEntry Y0-Y7
Reply with API	{@ none}

### Level 03 (Data Entry feature enabled) Readers

#### Y0-Y7: DATA ENTRY DATA

Data should be in ASCII, one character per byte. Data should be left justified (first character / key in Y0, second in Y1, etc.). The number of data bytes must equal eight (8) and unused data bytes must be sent as 00h.

If the data entry process is cancelled by the VMC for any reason, the VMC will send this message with all DATA ENTRY data bytes set to FFh.

Note: The cashless must translate the VMC key information into the appropriate key needed for the application

## 6.17. onRevalueRequest

A balance in the VMC account because coins or bills were accepted or some balance is left after a vend. With this event the VMC tries to transfer the balance to the payment media.

**Reply with API:** {@ sendRevalueApproved} or {@ sendRevalueDenied}

onRevalueRequest	byte[] data Revalue Amount Y0-Y1
Reply with AP	{@ sendRevalueApproved} or {@ sendRevalueDenied}

### Level 02/03 Readers

if "EXPANDED CURRENCY MODE" enabled

onRevalueRequest	byte[] data Revalue Amount Y0-Y3
Reply with AP	{@ sendRevalueApproved} or {@ sendRevalueDenied}

### Level 03 (EXPANDED CURRENCY MODE) Readers

Y0-Y1(or Y0-Y3): Revalue Amount - scaled.

The revalue amount should not exceed the revalue limit value given by the event

{@ onRevalueLimitRequest}



## 6.18. onRevalueLimitRequest

In a configuration with a bill and/or coin acceptor and payment media reader connected to a VMC, the VMC must know the maximum amount the payment media reader eventually will accept by a {@onRevalueRequest}. Especially if the bill acceptor accepts a wide range of bills. Otherwise the VMC may be confronted by the situation where it accepted a high value bill and is unable to pay back cash or revalue it to a payment media.

**Reply with API:** {@ sendRevalueLimitAmount}

onRevalueLimitRequest	none
Reply with API	{@sendRevalueLimitAmount}

**Level 02/03 Readers**

## 6.19. onSyncTimeDate

The VMC synchronize the Time/Date to the cashless.

**Reply with API:** {@ none}

onSyncTimeDate	byte[] timeDate Time Date Y0-Y9
Reply with API	{@none}

**Level 02/03 Readers**

Y0- Y9: Time/Date to synchronize the cashless device real time clock. The date bytes are BCD encoded.

Y0 = Years (Range: 00..99)

Y1 = Months (Range: 01..12)

Y2 = Days (Range: 01..31)

Y3 = Hours (Range: 00..23)

Y4 = Minutes (Range: 00..59)

Y5 = Seconds (Range: 00..59)

Y6 = Day of Week (Range: 01..07, Monday = 1..Sunday = 7)

Y7 = Week Number (Range: 01..53)

Y8 = Summertime (Range: 00..01, Summertime = 1)

Y9 = Holiday (Range: 00..01, Holiday = 1)

If any item of the time/date is not supported use FFH instead.

## 6.20. onDiagnostics

Device manufacturer specific instruction for implementing various manufacturing or test modes.

**Reply with API:** {@ sendDiagnosticsResponse}

onDiagnostics	byte[] data
	User Defined Data Y0-Yn
Reply with API	{@sendDiagnosticsResponse}

**Level 01/02/03 Readers**

Y0-Yn : User Defined Data.

The data portion of this command is defined by the manufacturer and is not part of this document.

## • 7. Cashless APIs

### 7.1. sendACK

After a {@ onVendFailure} event happened and If refunds success, the cashless should reply it with {@ sendACK}

sendACK	none
return	0 -> Success <0 -> Fail

**Level 01/02/03 Readers**

For Example:

```
if(refunded == true){  
    mCashlessManager.sendACK();  
}
```

### 7.2. sendJustReset

If the event {@ onReset} happened,the cashless device should reply it with {@ sendJustReset}.

sendJustReset	none
return	0 -> Success <0 -> Fail

#### Level 01/02/03 Readers

For Example:

```
mCashlessManager.sendJustReset();
```

## 7.3. sendTimeDateRequest

In certain circumstances it will be necessary to synchronize the real time clock of the cashless device with real time clock of the VMC. When the cashless device call this API and the VMC will send Date/Time by the event {@ onSyncTimeDate}

Usually, the developer can call this API after the event {@ onInitialComplete} come if it's necessary .

sendTimeDateRequest	none
return	0 -> Success <0 -> Fail

#### Level 02 / 03 Readers

For Example:

```
public int syncSystemTimeDate() {
    if(mCashlessInformation.FeatureLevel == 1){
        Log.e(TAG, "syncSystemTimeDate, level 1 not support this feature.");
        return ErrorCode.ERR_NO_SUPPORT;
    }
    return mCashlessManager.sendTimeDateRequest();
}
```

## 7.4. sendDisplayRequest

If the VMC have available display ,The cashless can request a message to be displayed on the VMC's display by call this API.

sendDisplayRequest	int displayTime	byte[] displayData
byte count	1 byte	DisplayRows*DisplayCols
return	0 -> Success <0 -> Fail	

#### Level 01/02/03 Readers

displayTime: Display Time - 0.1 second units

displayData: Display Data - ASCII

The message to be displayed. Formatting (leading and/or trailing blanks)is the responsibility of

the cashless.

NOTE: The number of bytes must equal the product of “DisplayRows” and “DisplayCols” up to a maximum of 32 bytes.

For Example:

```
public int displayRequest(int displayTime, byte[] displayData) {  
    if(mVmcInformation.DisplayRows == 0 || mVmcInformation.DisplayCols == 0){  
        Log.e(TAG, "VMC not support display");  
        return ErrorCode.ERR_FAIL;  
    }  
    return mCashlessManager.sendDisplayRequest(displayTime, displayData);  
}
```

## 7.5. sendBeginSession

Begin a vend session, Allow a patron to make a selection, but do not dispense product until funds are approved.

sendBeginSession	byte[] data
	Funds Available Z0-Z1
return	0 -> Success <0 -> Fail

### Level 01 Readers

sendBeginSession	byte[] data			
	Funds Available Z0-Z1	Payment media ID Z2-Z5	Payment Type Z6	Payment Data Z7-Z8
return	0 -> Success <0 -> Fail			

### Level 02 / 03 Readers

Z0-Z1: Funds Available – scaled

- Lesser of the user’s payment media or account balance or FFFEh units.
- Not yet determined - FFFFh. (Allows selection without displaying balance)

Z2-Z5: Payment media ID

00000000h-FFFFFFFEh=Payment media identification number.

FFFFFFFh= unknown payment media ID.

Z6: Type of payment:

00xxxxxb = normal vend card (refer EVA-DTS Standard, Appendix A.1.1 Definitions)

x1xxxxxb = test media

1xxxxxb = free vend card

xx00000b -0 VMC default prices

xx000001b -1 User Group(Z7 = EVA-DTS Element DA701)

Price list number(Z8 = EVA-DTS Element LA101)\*

xx000010b -2 User Group(Z7 = EVA-DTS Element DA701)

Discount group index (Z8 = EVA-DTS Element MA403)

xx000011b -3 Discount percentage factor (Z7=00, Z8 = 0 to 100\*\*,report as positive value in EVA-DTS Element MA404)

xx000100b -4 Surcharge percentage factor (Z7=00, Z8 = 0 to 100\*\*,report as negative value in EVA-DTS Element MA404)

\* User Group is a segmentation of all authorized users. It allows selective cost allocation. A User Group usually has no direct relation to a price list. Price Lists are tables of prices. Each Price List contains an individual price for each product. Discount Group indicates the Price List on which the Percentage Factor will be applied. If the User Group, the Price List or Discount Group is unknown by the VMC, the normal prices are used (Z6 is defaulted to 00h). Minimum value for Z7 and Z8 is 0.

\*\* Percentages are expressed in binary (00 to 64h) These functions may NOT be supported by all VMCs.

Z7-Z8 : Payment data as defined above.

If "EXPANDED CURRENCY MODE" enabled

sendBeginSession							
	Funds Available	Payment media ID	Payment Type	Payment Data	User Language	User Currency Code	Card Options
	Z0-Z3	Z4-Z7	Z8	Z9-Z10	Z11-Z12	Z13-Z14	Z15
return	0 -> Success <0 -> Fail						

### Level 03 (EXPANDED CURRENCY MODE) Readers

Z0-Z3: Funds Available - scaled

- Lesser of the user's payment media or account balance or FFFFFFFEh units.
- Not yet determined - FFFFFFFFh.

Z4-Z7: Payment media ID.

00000000h-FFFFFFFEh=Payment media identification number.

FFFFFFFh= unknown payment media ID.

Z8: Type of payment

Z9-Z10: Payment data as defined above.

Z11-Z12: User language to use during this session (2 ASCII characters per ISO639:latest version). The user language is read from the patrons card and, if supported, should be used instead of the VMC default language(taken according to the setup command International Telephone code) up to the next "session complete" . If the VMC is not able to support this language, the default setting should be used.

Z13-Z14: User currency code to use during this session per ISO 4217 (see Appendix A1). The value is configured as packed BCD with the leading digit a 1 (one). For example, the code for the US dollar would be 1840(Z13 = 18 and Z14 = 40). and for the Euro is 1978 (Z13 = 19 and Z14 =78).

Z15: Card options (overrides any previous default settings for reader)

- b0=0: The VMC displays the credit if it is programmed to do so
- b0=1: The VMC must not display the credit (privacy purpose - user option)
- b1=0: The actual inserted patrons card has no refund capability
- b1=1: The actual inserted patrons card has refund capability (Note: a reader with refund capability may

- be used with both type of cards)
- b2=0 The actual inserted patrons card has no revalue capability
- b2=1 The actual inserted patrons card has revalue & negative vend capability
- b3=0 The inserted patrons card has no partial refund capability
- b3=1 The inserted patrons card has partial refund capability
- b4-b7: Reserved for future extensions (unused bits must be set to 0)

For Example:

```
public int beginSession(){
    byte[] sendData = null;
    if(mCashlessInformation.FeatureLevel == 1){
        sendData = new byte[2];
        /** Z0-Z1 :Funds Available */
        sendData[0] = (byte)0xFF;
        sendData[1] = (byte)0xFF;
    }else if(mCashlessInformation.FeatureLevel == 2 || mCashlessInformation.FeatureLevel == 3){
        sendData = new byte[9];
        /** Z0-Z1 :Funds Available */
        sendData[0] = (byte)0xFF;
        sendData[1] = (byte)0xFF;
        /** Z2-Z5 :Payment mediaID */
        sendData[2] = (byte)0xFF;
        sendData[3] = (byte)0xFF;
        sendData[4] = (byte)0xFF;
        sendData[5] = (byte)0xFF;
        /** Z6 :Type of payment */
        sendData[6] = 0x00;
        /** Z7-Z8: Payment data */
        sendData[7] = 0;
        sendData[8] = 0;
    }
    return mCashlessManager.sendBeginSession(sendData);
}
```

## 7.6. sendSessionCancelRequest

The cashless is requesting the VMC to cancel the session. And than the VMC will initiate an event{@onSessionComplete}.This API is called when the VMC whenever the payment media is removed or a request for removal from the reader is made by the user (e.g. if a return button on the reader is pressed).

sendSessionCancelRequest	none
return	0 -> Success <0 -> Fail

## Level 01/02/03 Readers

For Example:

```
public int cancelSession(){  
    return mCashlessManager.sendSessionCancelRequest();  
}
```

## 7.7. sendVendApproved

Allow the selected product to be dispensed.

sendVendApproved	byte[] data
	Vend Amount Z0-Z1
return	0 -> Success <0 -> Fail

### Level 01/ 02/03 Readers

Z0-Z1: Vend Amount - scaled

This is the amount deducted from the user' s payment media or account.

This may not match the amount specified in the event {@ onVendRequest},it may be surcharged or discounted.

FFFFh - an electronic token was used.

If “basket / partial refund / options price” enabled

sendVendApproved	byte[] data	
	Vend Amount Z0-Z1 (Z0-Z3 @32bit)	Options Amount Z2-Z3 (Z4-Z7)
return	0 -> Success <0 -> Fail	

### Level 03 (“basket / partial refund / options price” enabled) readers

**Remark: “@32bit” means EXPANDED CURRENCY MODE enabled.**

Z0-Z1(or Z0-Z3): Vend Amount - scaled

Z2-Z3(or Z4-Z7): Options amount – scaled

This is amount deducted from the user' s payment media or account for options (part of vend amount which covers paid options). It is already included to Vend amount and just informs vending machine about discount / surcharge for options (for no discount / surcharge, it should be equal to “Options price” in the event{@onVendRequest }).

FFFFh - an electronic token was used.

For Example:

```
public int vendApproved(int vendAmount){  
    byte[] sendData = new byte[2];
```

```

    sendData[0] = (byte)(vendAmount >> 8);
    sendData[1] = (byte)(vendAmount & 0xFF);
    return mCashlessManager.sendVendApproved(sendData);
}

```

## 7.8. sendVendDenied

Approval denied for the patron's selection. Do not dispense any products.

sendVendDenied	none
return	0 -> Success <0 -> Fail

**Level 01/ 02/03 Readers**

For Example:

```

public int vendDenied(){
    return mCashlessManager.sendVendDenied();
}

```

## 7.9. sendMalfunctiontion

The cashless is reporting a malfunction or error.

After the event {@ onVendFailure} happened and if refunds failure cashless reply with this API.

sendMalfunctiontion	byte malfunctionCode
	Error Code Z0
return	-> Success <0 -> Fail

**Level 01/ 02/03 Readers**

Z0: Error Code - xxxxyyyy

xxxx error types

0000: Payment media Error1

0001: Invalid Payment media1

0010:Tamper Error1

0011:Manufacturer Defined Error1

0100:Communications Error2

0101:Reader Requires Service2

0110:Unassigned2

0111:Manufacturer Defined Error2

1000:Reader Failure3

1001:Communications Error3



1010:Payment media Jammed3  
1011:Manufacturer Defined Error  
1100:Refund error – internal reader credit lost  
1101-1111: Unassigned

For Example:

```
byte malfunctionCode = 0x00;  
mCashlessManager.sendMalfunctiontion(malfunctionCode);
```

## 7.10. sendRevalueApproved

A balance is in the VMC account because coins or bills were accepted or some balance is left after a vend. The VMC has issued an event {@onRevalueRequest} to the cashless device to transfer the balance to the payment media. The cashless accepted the request and added its value to the payment media balance. The cashless then reply with this API, so the VMC may clear the account.

sendRevalueApproved	none
return	-> Success <0 -> Fail

### Level 02/03 Readers

For Example:

```
/**  
 * to revalue,if complete reply {@sendRevalueApproved} , if not ok reply {@sendRevalueDenied}  
 */  
if(revalued){  
    mCashlessManager.sendRevalueApproved();  
}else{  
    mCashlessManager.sendRevalueDenied();  
}
```

## 7.11. sendRevalueDenied

A balance is in the VMC account because coins or bills were accepted or some balance is left after a vend. The VMC has issued an event {@onRevalueRequest} to the cashless to transfer the balance to the payment media. The cashless does not accept the request and reply with this API, so the VMC has to pay out change. It is a quite common situation if there is no payment media inserted at this moment.

sendRevalueDenied	none
return	-> Success <0 -> Fail

## Level 02/03 Readers

For Example:

```
/**
 * to revalue,if complete reply {@sendRevalueApproved} , if not ok reply {@sendRevalueDenied}
 */
if(revalued){
    mCashlessManager.sendRevalueApproved();
}else{
    mCashlessManager.sendRevalueDenied();
}
```

## 7.12. sendRevalueLimitAmount

The patron intends to revalue the payment media with a bill of some value. The VMC must know what kind of bills to accept, so it will issue an event {@onRevalueLimitRequest} to get the amount the cashless will accept. The cashless will reply with the scaled value,calculated with the maximum allowed payment media balance minus the current balance of the payment media. The cashless reply with {@sendRevalueDenied} if there is no payment media available upon this event.

sendRevalueLimitAmount	byte[] data Revalue Limit Amount Z0-Z1
return	-> Success <0 -> Fail

### Level 02 / 03 Readers

Z0-Z1 : Revalue limit value - scaled.

if “EXPANDED CURRENCY MODE” enabled.

sendRevalueLimitAmount	byte[] data Revalue Limit Amount Z0-Z3
return	-> Success <0 -> Fail

### Level 03 (EXPANDED CURRENCY MODE enabled) Readers

Z0-Z3 : Revalue limit value - scaled.

For Example:

```
if(revalueLimitAmount > 0){
    if(mExpanedCurrencyMode == false){
        byte responseData[] = new byte[2];
        responseData[1] = (byte)(revalueLimitAmount >> 8);
```

```

        responseData[2] = (byte)(revalueLimitAmount & 0xFF);
        mCashlessManager.sendRevalueLimitAmount(responseData);
    }else {
        byte responseData[] = new byte[4];
        responseData[1] = (byte)(revalueLimitAmount >> 24);
        responseData[2] = (byte)(revalueLimitAmount >> 16);
        responseData[3] = (byte)(revalueLimitAmount >> 8);
        responseData[4] = (byte)(revalueLimitAmount & 0xFF);
        mCashlessManager.sendRevalueLimitAmount(responseData);
    }
}
}

```

## 7.13. sendDataEntryRequest

The cashless is making a DATA ENTRY REQUEST.

sendDataEntryRequest	byte request
	Data Entry Length and Repeat Bit Z0
return	-> Success <0 -> Fail

### Level 03 Readers (if Data Entry option enabled)

Z0: DATA ENTRY LENGTH and REPEAT BIT

rnnnnnnnn

r – Repeat Bit (0 = initial request / 1 = repeated requests)

nnnnnnnn – number of requested characters / keys

For Example:

```

public int dataEntryRequest(byte entryData) {
    if(!mCashlessInformation.misDataEntryEnable){
        Log.e(TAG, "This feature not enabled.");
        return ErrorCode.ERR_NO_SUPPORT;
    }
    if(mVmcInformation.DisplayRows == 0 || mVmcInformation.DisplayCols == 0){
        Log.e(TAG, "VMC not support display");
        return ErrorCode.ERR_FAIL;
    }
    return mCashlessManager.sendDataEntryRequest(entryData);
}

```

## 7.14. sendDataEntryCancel

The user has pushed the reader's RETURN button before completing the DATA ENTRY. The VMC should terminate all DATA ENTRY activity in progress.

sendDataEntryCancel	none
return	-> Success <0 -> Fail

### Level 03 Readers (if Data Entry option enabled)

For Example:

```
public int dataEntryCancel() {  
    if(!mCashlessInformation.misDataEntryEnable){  
        Log.e(TAG, "This feature not enabled.");  
        return ErrorCode.ERR_NO_SUPPORT;  
    }  
    return mCashlessManager.sendDataEntryCancel();  
}
```

## 7.15. sendSelectionRequest

The cashless Requests to make a selection by VMC. The VMC should send event{@onVendRequest} to the same cashless device or {@onSelectionDenied}.

sendSelectionRequest	byte[] data					
	Funds Available Z0-Z1	Payment media ID Z2-Z5	Payment Type Z6	Payment Data Z7-Z8	Item Number Z9-Z10	Item Options Z11-Z14
return	-> Success <0 -> Fail					

### Level 03 (remote vend enabled) Readers

Z0-Z1: Funds Available – scaled (maximum allowed cost which could be credited from the cashless device). FFFFh means undefined credit / coupon (whole product cost should be credited from the cashless device). 0000h requests to dispense using cash (or another credit sources).

Z2-Z5: Payment media ID. FFFFFFFFh = unknown payment media ID.

Z6: Payment type

Z7-Z8: Payment data

Z9-Z10: The item number to be dispensed. Item number is defined by manufacturer and equals to item number of {@ onVendRequest}

Z11-Z14: Item options

b0: 1 = not last vend in basket. Means that after dispensing this product, the cashless device is going to

request another one. Vending machine which has receptacle could skip receptacle opening to speedup the process. If receptacle was not opened and the cashless device did not request another vend during 5 seconds, machine should open receptacle.

b1: 1 = no lid for coffee machines with lids dispensing. Other machines should ignore this bit.

b4-b2: sugar for coffee machines. Other machines should ignore these bits. 000 = default, 001 = no sugar, 010-110= intermediate, 111 = max.sugar.

b6-b5: cup size. 00 = default, 01 - smallest, 11 - largest.

b9-b7: beverage strength. 000 = default, 001 - smallest, 111 - largest.

b12-b10: option 1 amount (whitener / taste strength). 000 = default, 001 - no / min, 111 - max.

b15-b13: option 2 type (syrup number / taste 2 type). 000 = no, 001-111 - 7 different types.

b18-b16: option 2 amount. 000 = default, 001 = min, 111 = max.

b31-b19: reserved and should be 0.

If "Expanded Currency Mode" enabled

sendSelectionRequest	byte[] data								
	Funds Available Z0-Z3	Payment media ID Z4-Z7	Payment Type Z8	Payment Data Z9-Z10	User Language Z11-Z12	User Currency Code Z13-Z14	Card Option Z15	Item Number Z16-Z17	Item Options Z18-Z21
retrun	-> Success <0 -> Fail								

### Level 03 (remote vend and Expanded Currency Mode enabled) Readers

Z0-Z3: Funds Available - scaled (maximum allowed cost which could be credited from the cashless device). FFFFFFFFh means undefined credit / coupon (whole product cost should be credited from the cashless device). 00000000h requests to dispense using cash (or another credit sources).

Z4-Z7: Payment media ID. FFFFFFFFh = unknown payment media ID.

Z8: Payment type

Z9-Z10: Payment data

Z11-Z12: User language to use during this vend.

Z13-Z14: User currency code.

Z15: Payment media options.

b0: Reserved for future extensions (unused bits must be set to 0)

b1=0: The media has no refund capability

b1=1: The media has refund capability

b2: Reserved for future extensions (unused bits must be set to 0)

b3=0 The media has no partial refund capability

b3=1 The media has partial refund capability

b4-b7: Reserved for future extensions (unused bits must be set to 0)

Z16-Z17: The item number to be dispensed.

Z18-Z21: Item options

For Example:

```
return mCashlessManager.sendSelectionRequest(sendData);
```

## 7.16. sendCouponReport

Informs machine that a coupon was accepted.. Once coupon is reported to machine, it is not possible to report another one before followed “{@ onCouponReply}” event from VMC or before it will be reset.

sendCouponReport	byte[] data			
	Item Number Z0-Z1	Coupon Options Z2	Coupon Value Z3-Z4 (Z3-Z6 @32bit)	Coupon ID Z5-Z8 (Z7-Z10 @32bit)
return	-> Success <0 -> Fail			

### Level 03 (coupons feature enabled) Readers

**Remark: “@32bit” means EXPANDED CURRENCY MODE enabled.**

Z0-Z1: Item Number to be sponsored. Coupon value may be used only for reported Item Number. FFFFh may be interpreted as “no dedicated selection” , coupon is for free use.

Z2: Coupon Options:

Bit 0 set: Allow no other credit source (from reporting cashless) for the rest of product price. Coupon should be denied if there is already other credit,all other payment sources will be disabled during Coupon sequence active. VMC will not dispense without VEND REQUEST / VEND APPROVED even if coupon value meets product price.

Bit 1 set: Machine may dispense product immediately if there is enough credit.

Bit 2 ... 7: reserved for future use, should be set to 0.

Z3-Z4(or Z3-Z6):Coupon value, scaled. Should be handled similar to a coin value token.Value 0xFFFFh reports a vend coupon (vend token), worth one vend.Coupons may be used for one vend only even if coupon value is morethan product price. Coupon value must not be returned as cash. In mixed payed vends machine will first use coupon, then cash and at last cashless.

Z5-Z8(or Z7-Z10): Coupon ID, may be used for billing purposes. Sent as unsigned integer 32,MSB first.

## 7.17. sendDiagnosticsResponse

The data portion of this response is defined by the manufacturer and is not part of this document.

sendDiagnosticsResponse	byte[] data
	User Defined Z0-Zn
return	-> Success <0 -> Fail

## Level 01/02/03 Readers

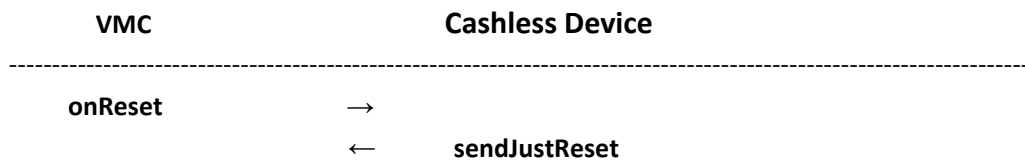
Z0-Zn: User Defined Data.

### • 8. Return Code

Return code	value	description
success	0	successfully
err_fail	-1	failure
err_param	-2	parameter error
err_un_start	-3	cashless not start
err_mem	-4	memory error
err_write	-5	mdb port send error
err_read	-6	mdb port receive error
err_timeout	-7	receive timeout
err_no_support	-8	no support
err_out_of_sequence	-9	out of sequence

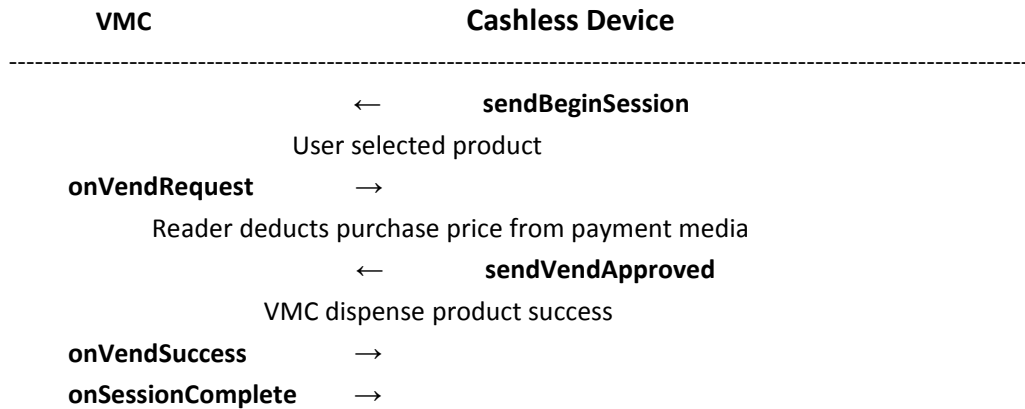
### • 9. Appendix

#### 9.1. Appendix1 Reset Sequence

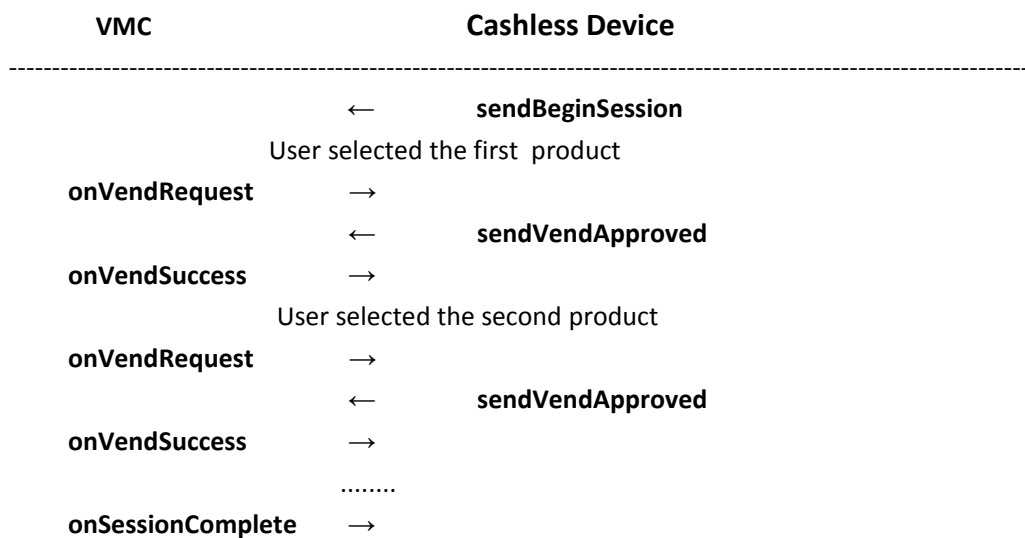


## 9.2. Appendix2 Vend Sequence

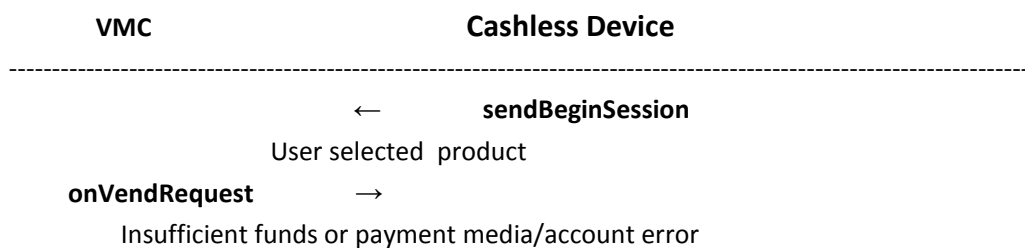
### A: Vend Approved for Single Vend



### B: Vend Approved for Multiple Vend



### C: Vend denied





	←	<b>sendVendDenied</b>
<b>onSessionComplete</b>	→	

**D: Session cancelled by user with reader return button**

<b>VMC</b>		<b>Cashless Device</b>
<hr/>		
	←	<b>sendBeginSession</b>
		User pushes reader RETURN button
	←	<b>sendSessionCancelRequest</b>
<b>onSessionComplete</b>	→	

**E: Session cancelled by user via coin mechanism escrow return button before product was selected**

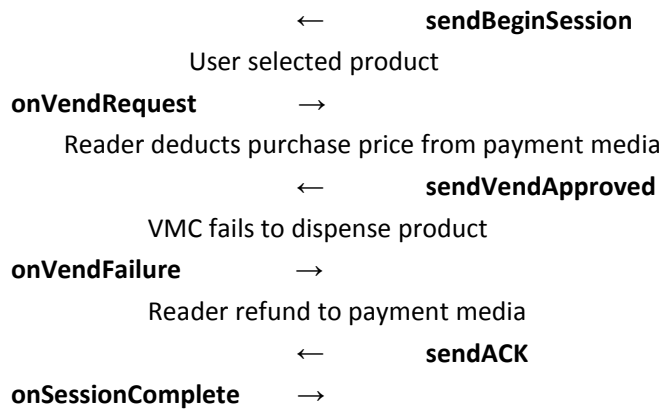
<b>VMC</b>		<b>Cashless Device</b>
<hr/>		
	←	<b>sendBeginSession</b>
		User pushes coin mech. escrow return
<b>onSessionComplete</b>	→	

**F: Session cancelled by user via coin mechanism escrow return button after product was selected**

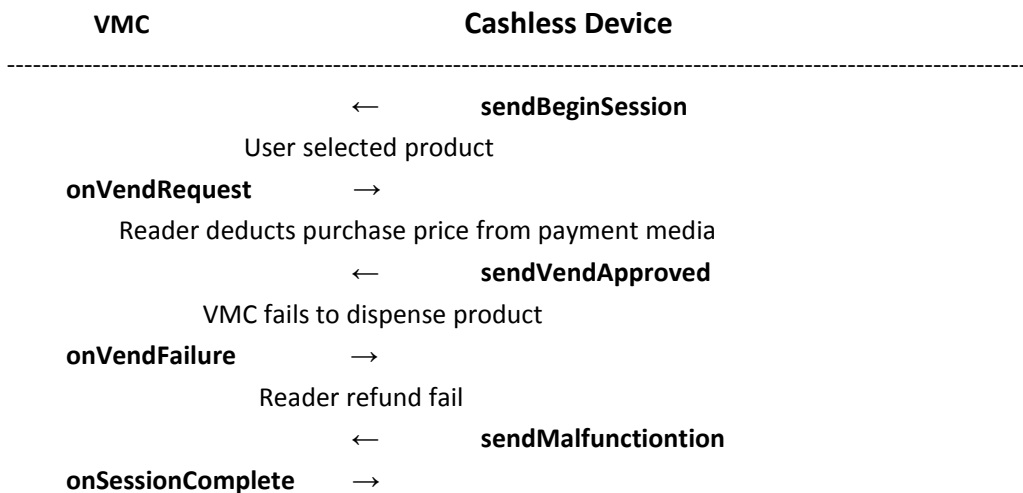
<b>VMC</b>		<b>Cashless Device</b>
<hr/>		
	←	<b>sendBeginSession</b>
		User selected product
<b>onVendRequest</b>	→	
		User pushes coin mech. escrow return
<b>onVendCancel</b>	→	
	←	<b>sendVendDenied</b>
<b>onSessionComplete</b>	→	

**G: Vend Failure/product not dispensed/refund success**

<b>VMC</b>		<b>Cashless Device</b>
<hr/>		

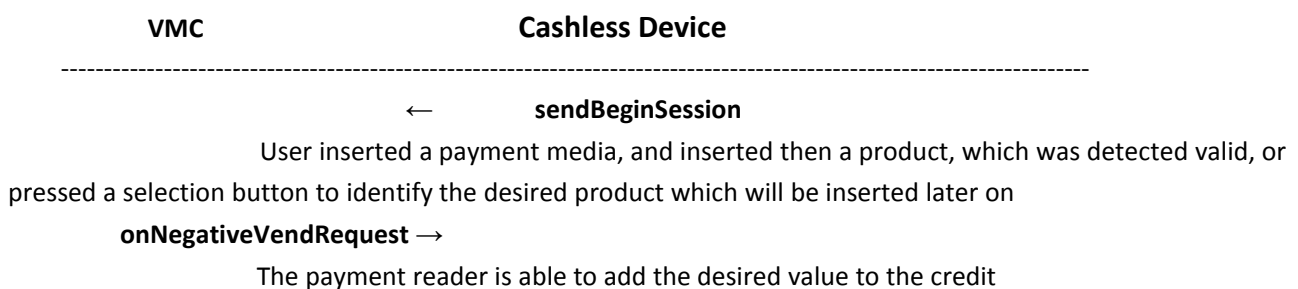


#### H: Vend Failure/product not dispensed/refund failed



## 9.3. Appendix3 Negative Vend Sequence

#### A: Negative Vend Approved for Single Vend



← **sendVendApproved**

The product is now fully accepted from the machine or the user has finally finished insertion of a valid product

**onVendSuccess** →

**onSessionComplete** →