Technical Documentation

NRI
A Crane Co.Company

Payment Manager

NRI Payment Manager

for the comfortable vending control between PC and NRI/CashCode payment systems

Instructions to implement the paymentmanager.dll

04.08 Goe/Hns/DAl/ds/vBi
Edition 2.0
AB.PM-GB

CRANE

National Rejectors, Inc. GmbH • Zum Fruchthof 6 • D-21614 Buxtehude
Phone: +49 (0)4161/729-0 • Fax: +49 (0)4161/729-115 • E-mail: info@nri.de • Internet: www.nri24.com

Table of contents

1 Introduction

This manual deals with implementing and applying the NRI program library “Payment Manager” (paymentmanager.dll).

1.1 What’s the use of the NRI Payment Manager?

The NRI Payment Manager has been specifically developed to meet the needs of PC-controlled vending, service, ticketing or amusement machines that have to handle NRI and CashCode payment units, such as coin or banknote validation/payout systems:

- Coin changers, e.g. NRI C^2 MDB, E-66/A-66 MDB or NRI G-46 MDB with USB or RS-232 interface
- Coin validators, e.g. NRI G-13.mft ccTalk/MDB/S1, NRI G-18.mft MDB/S1, NRI G-40 ccTalk/S1
- Bill validators
- Hoppers, e.g. NRI G-38.1xxx

The comfortable NRI Payment Manager undertakes all tasks of the vending control such as the communication between the payment units connected and the PC irrespective of the PC interface (USB or RS-232) or the data transfer protocol for the payment units.

1.2 The chapter contents

Chapter 2 presents the NRI Payment Manager and its five library functions “Openpaymentmangager”, “Closepaymentmanager”, “Startpaymentmanager”, “Stoppaymentmanager”, and “Setpaymentmanager” as well as the paymentmanager.ini for additional setup information.

How to handle events such as cash acceptance and payout, new device statuses or errors using the windows message system, is defined in Chapter 3.

Chapter 4 "Example" explains the necessary steps for starting the NRI Payment Manager using the Openpaymentmanager, Startpaymentmanager, and Setpaymentmanager functions, as well as for calling all coin/banknote denominations in use and paying out cash using the Setpaymentmanager.

The example code in Chapter 5 helps you applying the NRI Payment Manager.

The payment system terminology used in this manual is defined in Chapter 6 which, along with Chapter 7 "Quick Reference", shortens the search for specifc explanations.

2 Library functions/additional setup information (paymentmanager.ini)

The NRI Payment Manager is a library (paymentmanager.dll) that provides five functions, which can be applied by the user. First of all the DLL module must be loaded during runtime, so that the function addresses can be determined in a next step. The following functions can be used:

- Openpaymentmanager
- Closepaymentmanager
- Startpaymentmanager
- Stoppaymentmanager
- Setpaymentmanager

However, the user must not only load the DLL, but also has to handle Windows messages sent by the NRI Payment Manager. This enables the NRI Payment Manager to post the events of the relevant payment units to the user. To be able to get these messages the user must start the NRI Payment Manager with the information of his window handle and the address of the Windows message.

2.1 openpaymentmanager

The Openpaymentmanager function sets up communication between the NRI Payment Manager and the PC interface of the payment units (USB or RS-232). When calling this function the NRI Payment Manager tests, if it is ready to communicate with the payment units. This is the first function for the user to start the NRI Payment Manager. Only if the user is able to open the NRI Payment Manager, can the NRI Payment Manager be started.

```
int       openpaymentmanager(void);
```

Parameter:

port

Communication ports to search for devices:

USB only	=	0x10000
COM1 only	=	0x00101
COM1..15	=	0x0010F

Return value:

If the function was executed successfully, the return value is “0”.
If the function failed, the return value is higher than “0” (error code).

Error codes:

Return value	*Meaning*
1000	No USB DLL (NRIHiDAPI.DLL) found

2.2 closepaymentmanager

The Closepaymentmanager function makes the NRI Payment Manager close the communication handle with the payment units.

```
int      closepaymentmanager(void);
```

Parameters:

None

Return value:
If the function was executed successfully: 0.
If the function failed: 1

2.3 startpaymentmanager

The Startpaymentmanager function initiates the NRI Payment Manager. When calling this function the NRI Payment Manager tries to find the payment units connected. It first of all initializes the payment units and then keeps them working. The user has to give the handle to his window and post the address, where to send the window messages when an event occurs, to the NRI Payment Manager.
The user can determine the

- payment unit the NRI Payment Manager is to address,
- message system,
- machine interface.

By setting these parameters to “0”, the NRI Payment Manager tries to enable all connected payment units with the known machine interfaces and sends the default Windows messages.

```
int     startpaymentmanager(
        HWND windowsHandle,          // Handle to the user window
        int messageAddress,          // Address of the Windows messages
        int devices,                 // Payment units to be enabled
        int messageConfig,           // Kind of message
        int protocol                 // Machine interfaces to be enabled
        );
```

Parameters:

windowsHandle

Gives the NRI Payment Manager the handle to the user application/window.

messageAddress

Address of the Windows messages sent by the NRI Payment Manager. This address must be specified by the user.

Note:

The address must be higher than the WM_USER address given by Windows.

Example: `#define WM_PAYMENTMESSAGE (WM_USER+1)`

devices

Using this parameter the user is able to select the payment units. If it is set to “0” the NRI Payment Manager enables all found payment devices. If some of the payment units connected are not to be enabled, the user can select the following payment units bitwisely:

bits	*b4*	*b3*	*b2*	*b1*	*b0*
	reserved for future application	0: hopper off 1: hopper on	0: cashless system off 1: cashless system on	0: bill validator off 1: bill validator on	0: coin changer/ validator off 1: coin changer/ validator on

Example: If only the hopper and the coin changer/validator shall be enabled, the *devices* parameter has to be 1001 b = 5 d. 0000 b = 0d means all devices are enabled.

Hopper (s) must be defined in the paymentmanager.ini file (s. section 2.6 "Additional setup information (paymentmanager.ini)").

messageConfig

Using this parameter the user can configure the messages sent by the NRI Payment Manager. If the parameter is set to “0”, the NRI Payment Manager uses the default Windows message. Other parameter values than “0” are reserved for future application.

protocol

Using this parameter the user can select the machine interface. If it is set to “0”, the NRI Payment Manager enables all machine interfaces known. If not all protocols are to be enabled, the user can select the following machine interfaces bitwisely:

bits	***b4***	***b3***	***b2***	***b1***	***b0***
	reserved for future application	reserved for future application	0: CCNet off 1: CCNet on	0: ccTalk off 1: ccTalk on	0: MDB/S1 off 1: MDB/S1 on

Example: If only the MDB/S1 protocol is supposed to be enabled, the *protocol* parameter has to be 0001 b = 1 d.

Return value:

The payment units found are sent bitwisely (cp. parameter *devices*). Return values higher than or equal to “0x2000” are error codes.

Return value	***Meaning***
0	No payment units found
1	Coin changer/validator found
2	Bill validator found
3	Coin changer/validator and bill validator found
4	Cashless payment system found
8	Hopper(s) found
16	Escrow(s) found
32	Display found
0x2000	NRI Payment Manager is not open (Openpaymentmanager( ) first)
0x2001	Unknown protocol selected
0x2002	Error in MDB/USB adapter communication
0x2003	Error in MDB/USB adapter communication
0x2004	No communication with payment units
0x2005	Payment Manager is already running
0x2006	Payment Manager could not start thread

Notes:
For normal applications the user just has to send Startpaymentmanager (*windowsHandle*, *messageAddress*,0,0,0) to start the NRI Payment Manager.

2.4 stoppaymentmanager

The Stoppaymentmanager function enables the NRI Payment Manager to cease communication with the payment units.

```
int       stoppaymentmanager(void);
```

Parameters:

None

Return value:
If the function was executed successfully, the return value is “0”.
If the function failed, the return value is higher than “0”.

2.5 setpaymentmanager

The Setpaymentmanager function enables the user to

- enable/disable certain coins or bills
- payout a certain amount of money
- poll status information from the payout units
- temporarily depose certain banknotes in the escrow and not transport them straightly to the stacker.

After starting the NRI Payment Manager all coins and bills are disabled, so that the user has to enable them by using the Setpaymentmanager function. The easiest way to do so is to call this function with all parameters set to “0”. This means that all means of payment are enabled.

```
unsigned long int    setpaymentmanager(
                     int command,         // Command for NRI Payment Manager
                     int selection,       // Selection for command
                     int info 1,          // Information for selection, if necessary
                     int info 2           // Information for selection, if necessary
                     );
```

Parameters:

command

Command code for the NRI Payment Manager to carry out the action required.

Command	*Meaning*
0	Enable/disable
1	Payout
2	Status
3	Escrow
4	Sorting device
5	Display
6	Maintenance

selection

Using this parameter the user can specify the *command* parameter (see overview table).

info1

Information for the NRI Payment Manager concerning the command selection (if necessary, see overview table).

info2

Information for the NRI Payment Manager concerning the command selection (if necessary, see overview table).

Return value:
The Return Value is a reply to the selected command given by the NRI Payment Manager.

2.5.1 Overview table of Setpaymentmanager commands

Command	*Selection*	*Info1*	*Info2*	*Return Value*	*Meaning*
0 "Inhibit"	**0**	**–**	**x**	A*	Enable all means of payment
	1	**–**	**x**	A*	Disable all means of payment
	2	**denomination**	**x**	A*	Enable denomination selected
	3	**denomination**	**x**	A*	Disable denomination selected
	4	**denomination ID**	**x**	A*	Enable denomination with ID selected
	5	**denomination ID**	**x**	A*	Disable denomination with ID selected
1 "Payout"	**0**	**amount**	**x**	Value paid out	Try to pay out amount selected and reply with value actually paid out (payment unit selected automatically)
	1	**amount**	**pay-ment unit**	Value paid out	Try to pay out amount selected by payment unit selected and reply with value actually paid out
	2	**x**	**x**	Lowest value that can be paid out	Reply with lowest amount possible that can be paid out
	3	**x**	**x**	Highest value that can be paid out	Reply with highest amount possible that can be paid out
	4	**payment unit**	**pay ment unit no.**	Value	Reply with payout item(s) of payout unit selected (define unit no., if there is more than one unit connected of a certain type of payout unit (2 hoppers)

Command	*Selection*	*Info1*	*Info2*	*Return Value*	*Meaning*
1 "Payout"	**6**	**payment unit**	**x**	Bit mask	Reply with empty/full status of unit selected bits 7..0: bit 5 = high level sensor supported bit 4 = low level sensor supported bit 1 = high level reached (unit full) bit 0 = low level underrun (unit empty)
	7	**payment unit**	**en/dis**	A*	Enable/disable unit selected for automatic payout (1=en, 0=dis), after start-up all units are enabled
	8	**value**	**num**	A*	Set number of bills to be payed out from bill-to-bill unit (e.g. value = 500, num = 2 for payout of two 5-euro bills)
	8	**0**	**0**	A*	Pay out bills from bill-to-bill unit set with previous command (first set value for each denomination and then start payout)
2 "Status"	**0**	**denomination ID**	**x**	Denomination of means of payment: -1 = item does not exist	Reply with the denomination of the ID selected
	1	**denomination ID**		Payment unit: 1 = coin changer/ validator 2 = bill validator -1 = item does not exist	Reply with payment unit the denomination is programmed in

Command	*Selection*	*Info1*	*Info2*	*Return Value*	*Meaning*
2 "Status"	**2**	**denomination ID**	**x**	Cash collector: 0 = cash-box 1 = tube -1 = item does not exist	Reply with unit the denomination selected is collected in
	3	**denomination ID**	**x**	Availability status: 0 = disabled 1 = enabled -1 = item does not exist	Reply, whether denomination selected is enabled or disabled
	4	**denomination ID**	**x**	Availability status: Number of coins in tube -1 = item does not exist	Reply with number of coins in respective tube of denomination selected
	5	**denomination ID**	**x**	Sorting status: Number of sorter path	Reply with cash target/sorter path programmed for denomination selected
3 "Escrow"	**0**	**x**	**0/1**	A*	Escrow off (0)/on (1) for all bills
	1	**denomination ID**	**0/1**	A*	Escrow off (0)/on (1) for bill/coin selected
	3	**x**	**1/2**	A*	Accept (1)/return (2) bill from escrow
	3	**x**	**3**	A*	Extend timeout
	4	**escrow number (0 = all)**	**1/2**	A*	Open coin escrow with number selected and accept (1) or reject (2) coins
	5	**x**	**x**	1 = opened 0 = closed	Reply with escrow status

Command	*Selection*	*Info1*	*Info2*	*Return Value*	*Meaning*
4 "Sorting device" (see section 2.5.2)	**0**	**denomination ID**	**path 1..8**	0 = OK -1 = item does not exist	Set sorter/override path(s) selected for denomination selected
	1	**sorter override**	**x**	0 = OK	Set override mask (sorter paths to be used for sorting/not to be used for sorting)
	2	**bill cassette ID**	**value**	A*	Set value selected for denomination collected in cassette selected
	3	**x**	**x**	Status	Reply with status of sorting device additionally connected
5 "Display"	**0**	**0**	**0**	Size status	Reply with number of display lines and characters (0x0110 = 1x16 = 1 line with 16 characters)
	1	**mode (0 = solid text, the only mode available so far)**	**char string**	0 = OK	Set string of characters (pointer to text) for solid text
6 "Maintenance"	**0**	**bill cassette no.**	**num**	num unloaded	Unload number of banknotes selected from bill-to-bill recycling cassette number selected and reply with number effectively unloaded

*	Return value "A":	≥ 0	= OK with value returned
		-1	= cash ID not found, device not found, execution error
		-2	= value too small
		-3	= no device attached
		-4	= device error
		-5	= command unknown/not supported
		-6	= Payment Manager is not running
x	of no importance		

2.5.2 Explanations for coin validators with sorting control

If the Payment Manager detects a coin validator with sorting capability, the Payment Manager tries to configure sorting automatically. The necessary information is expected to be in the "paymentmanager.ini" file (s. Chap. 6 "Additional setup information (paymentmanager.ini)". If this file does not exist or does not contain all information required, the sorting will not be configured. Using the Setpaymentmanager command it is possible to configure the coin sorting directly and/or to modify coin sorting during runtime (e.g. as hopper is full, coins should be directed to cash-box). Depending on the validator two sorting modes are possible:

a) Single sorter path/coin without override (G-13.mft)
For each coin, one sorter path 1…8 can be set at any time.
Example:
Setpaymentmanager(4,0,3,6), i.e. coin with ID 3 is sorted using sorter path 6

If the payout unit of the set sorter path is full, coins will be sorted into the cash-box (factory-made cash-box path).

b) Multiple paths/coin with override (G-13.mft, G-40)
For each coin, four sorter paths 1...8 can be set during configuration. During runtime only the override mask may be changed.
Example:
Setpaymentmanager(4,0,3,0x1345), i.e. regular sorter path and three override sorter paths for coin with ID 3 = 0x1345 (hex):

with ID 3 = 0x1345 (hex):	regular sorter path = 5
	1st override path = 4 (if payout unit of path 5 is full)
	2nd override path = 3 (if payout units of paths 5, 4 are full)
	3rd override path = 1 (if payout units of paths 5, 4, 3 are full)

If all four payout units of the set sorter/override paths are full, coins will be sorted into the cash-box (factory-made cash-box path).

The override mask specifies which sorter paths are no longer to be used to sort the coins:

Bit 7…0 = sorter path 8..1:	bit set (1): sorter path is to be used for sorting (payout unit not full)
	bit not set (0): sorter path is not to be used for sorting (payout unit full)

2.6 Additional setup information (paymentmanager.ini)

The paymentmanager.ini contains additional setup information the Payment Manager needs to work with hoppers and sorters. I.e. the Payment Manager will only search for hoppers that are defined in the ini file. The following sections list what must be defined.

2.6.1 Hopper definition

Format:	-H,address,coin,path
Example:	-H,3,50,5
Explanation:	Hopper with address 3 pays out 50-cent coins and can be reached via sorter path 5

☞ Do not assign sorter paths also defined for sorting devices (s. section 2.6.3 "Sorting device definition").

2.6.2 Cash-box definition

Format:	-C,path
Example:	-C,1
Explanation:	Cash-box is reached through sorter path 1

2.6.3 Sorting device definition

Format:	-S,coin,path
Example:	-S,20,3
Explanation:	20-cent coins will be routed to sorter path 3

☞ Do not assign sorter paths also defined for hoppers (s. section 2.6.1 "Hopper definition").

2.6.4 Bill recycling cassette definition for bill-to-bill unit

Format:	-B,bill,cassette number
Example:	-B,500,1
Explanation:	5-euro bills will be stored in bill-to-bill cassette 1

2.6.5 Escrow definition

The escrow must be defined to suppress the "opened" status for escrows not installed.

Format:	-E,connector number
Example:	-E,1
Explanation:	Escrow 1 is installed

2.6.6 Definition for additional status messages

The following status messages can be defined till now:

- "Bill2Bill routing information"
- "ccTalk currency code"
- "Coins payed out by coin changer"

(Cp. section 3.2 "Iparam")

Format:	-A,message
Example:	-A,1
Explanation:	The "Bill2Bill routing information" message will be issued

3 The NRI Payment Manager Windows message system

The NRI Payment Manager must inform the user about special events such as e.g.:

- cash acceptance,
- cash payout,
- new device statuses,
- escrow events or
- errors.

For this the NRI Payment Manager uses the Windows Send Message function.
Using the Startpaymentmanager function the user gives the handle to the user window and the address selected for the messages to the NRI Payment Manager. The user is thus responsible for handling all messages sent by the NRI Payment Manager, as the Manager sends a message to the window and does not return until the window procedure has processed the message. Therefore the user's message routine has to return a result as fast as possible.

The messages sent by the NRI Payment Manager include two parameters: wparam and lparam. These two parameters inform the user about an event.
The first parameter "wparam" contains information about the event and from which payment unit the event has been sent. The second parameter "lparam" gives more detailed information about the event indicated by the wparam, such as concrete values, status or error codes.

3.1 wparam

bits 15,14,13,12	*bits 11,10,9,8*	*bits 7,6,5,4 = Event*	*bits 3,2,1,0 = Payment unit*
reserved for future application	reserved for future application	**0 = Status of payment unit** **1 = Cash acceptance** **2 = Cash payout (manual)** **3 = Escrow/escrow lever activation** **4 = Error**	**0 = Payment Manager DLL** **1 = Coin changer/validator** **2 = Bill validator** **3 = Cashless system** **4 = Hopper**

3.2 lparam

Event of wparam	*Meaning of lparam*
Status (0)	**0 = Found (during start-up)** **1 = Ready** **2 = Out of order (hardware problem)** **3 = Not found (while running)** **4 = Tube of coin changer empty** **5 = Not found (during start-up)** **6 = Reconnected** **7 = Bills removed from dispenser** Additional status messages (cp. section 2.6.6): 0x0000 001y = Accepted bill routed to cassette no. y (0 = drop cassette, for bill-to-bill only) 0x0000 1xyy = Payed out no. yy of coin ID x (MDB coin changers only) 0x0001 yyyy = Accepted currency code (ccTalk coin acceptance only)
Cash acceptance (1)	**Denomination**
Cash payout (2)	**Denomination**
Escrow (3)	**0 = Return lever activated** **> 0 = Denomination routed to escrow position (value in escrow position increased)**
Error (4)	**Code**

Examples:

1) A 1,00 $ banknote has been inserted into a bill validator:
NRI Payment Manager message: “cash acceptance” from “bill validator” (wparam 12 hex. = 18 dec.) and “denomination” = “100” (lparam 100 dec.).

2) 0,25 $ have been paid out by the coin changer manually:
NRI Payment Manager message: “cash payout” from “coin changer” (wparam is 21 hex. = 33 dec.) and “denomination” = “25” (lparam 25 dec.).

3) A hopper error occurs:
NRI Payment Manager message: “error” in “hopper” (wparam is 44 hex. = 68 dec.) with relevant “code” (lparam).

Error codes:

lparam error code	*Meaning*
0	Communication error
1	Coin changer/validator reset
2	Bill validator reset
3	Sensor problem
4	Defective coin changer motor
5	Coin jam in changer tube
6	Coin jam in validator
7	Bill validator ROM checksum error
8	Coin changer/validator ROM checksum error
9	Bill validator cash-box out of position
10	Defective tube sensor
11	Payment unit disabled
12	Validator unplugged
13	Coin jam
14	Coin sorting error
15	String recognition (coin inserted on a string)

4 Examples

4.1 Starting the NRI Payment Manager

The easiest way to start the NRI Payment Manager is to use the functions:

- Openpaymentmanager
- Startpaymentmanager
- Setpaymentmanager

1. Open the NRI Payment Manager using the Openpaymentmanager function.
   The manager is ready.
2. Start the NRI Payment Manager using the Startpaymentmanager function with the last three parameters set to "0".
   The NRI Payment Manager will address and initialize all payment units connected irrespective of the machine interface. However, they will still be disabled.
3. Enable all payment units connected and all denominations programmed in the payment units using the Setpaymentmanager function with all parameters set to "0".

Note:
If you do not want to enable all payment units, set the relevant parameters to different values to disable the required units and denominations (see Chap. 2.5).

Step	*Function (parameters)*	*Return value*	*Meaning*
1	**Openpaymentmanager( )**	**0**	**NRI Payment Manager is open and ready.**
2	**Startpaymentmanager(this->m_hWnd, WM_PAYMENTMESSAGE, 0,0,0)**	**Units found (e.g: 1 = changer, see Chap. 2.3)**	**NRI Payment Manager starts, all connected and known units found, handled and initialized by the manager.**
3	**Setpaymentmanager(0,0,0,0)**	**0**	**All payment units with all programmed denominations enabled.**

4.2 Which denominations can be used?

The user can get any status information about the denominations programmed in the payment units using the Setpaymentmanager function.
To find out which coins and banknotes are available:

1. Set parameter 1 to "2".
   Command "status" is selected.

2. Set parameter 2 to “0”.
   Selection for command is specified: poll denomination of ID selected using the next parameter.
3. Set parameter 3 (info1) to “0”–“x”.
   Denomination of ID “0”–“x” is polled.
   Note:
   If the NRI Payment Manager replies “0”, the denomination ID selected does not exist.

Example:
If you want to know the denomination of ID 0, use function Setpaymentmanager(2,0,0,0), and the payment unit the coin/banknote is programmed in will return the coin/banknote denomination.

The following table lists the standard coin/banknote programming of US changers and bill validators:

Function (parameters)	*Return value*	*Meaning*
Setpaymentmanager(2,0,0,0)	**5**	**ID 0 = 0.05 $ (coin)**
Setpaymentmanager(2,0,1,0)	**10**	**ID 1 = 0.10 $ (coin)**
Setpaymentmanager(2,0,2,0)	**25**	**ID 2 = 0.25 $ (coin)**
Setpaymentmanager(2,0,3,0)	**50**	**ID 3 = 0.50 $ (coin)**
Setpaymentmanager(2,0,4,0)	**100**	**ID 4 = 1.00 $ (banknote)**
Setpaymentmanager(2,0,5,0)	**500**	**ID 5 = 5.00 $ (banknote)**
Setpaymentmanager(2,0,6,0)	**-1**	**ID 6 does not exist**

4.3 Paying out cash

Paying out cash is very easy. Use again the Setpaymentmanager function:

1. Set parameter 1 to “1”.
   Command “payout” is selected.
2. Set parameter 2 to “0”.
   Selection for command is specified: payout amount selected using the next parameter.
3. Set amount of money to be paid out in parameter 3 (info1).
   The payment unit(s) will try to pay out the amount selected and the NRI Payment Manager will reply the amount actually paid out.

Example:
If you would like to pay out 1.25 $, use function Setpaymentmanager(1,0,125,0).

Note:
If the return value of the function is only 100 (1.00 $) and not 125 (1.25 $), the payment unit(s) has/have not enough change to pay out the whole amount. In this case you could poll the lowest and highest amount, that can be paid out:

1. Set parameter 1 to “1”.
   Command “payout” is selected.
2. Set parameter 2 to “2” for the lowest amount possible or to “3” for the highest amount possible.
   Selection for command is specified: NRI payment manager will poll the lowest/highest amount, that can be paid out.

Function (parameters)	*Return value*	*Meaning*
Setpaymentmanager(1,2,0,0)	**5**	**Lowest payout value is 0.05 $**
Setpaymentmanager(1,3,0,0)	**100**	**Highest payout value is 1.00 $**
Setpaymentmanager(1,0,125,0)	**100**	**Payout command: 1.25 $, but only 1.00 $ can be paid out**

4.4 Monitoring the credit

Information about cash insertion into the payment units is directly sent to the user as event by the NRI Payment Manager using Windows messages, as well as cash payout events. The two parameters “wparam” and “lparam” of the messages indicate, what denomination has been inserted into or paid out by which payment unit (see Chap. 3.1 and 3.2).

The following table lists examples demonstrating the meaning and possible consequences of the messages:

Event	*wparam, lparam*	*Meaning -> possible consequence*	*Credit*
–	**–**	**start**	**0.00 $**
1.00 $ inserted in coin changer/ validator	**17, 100**	**wparam = 17d = 11hex = cash acceptance in coin changer/validator** **lparam = 100d = 1.00 $ denomination**	**1.00 $**
20.00 $ inserted in bill validator	**18, 2000**	**wparam = 18d = 12hex = cash acceptance in bill validator** **lparam = 2000d = 20.00 $ denomination**	**21.00 $**
Coin changer return lever pressed	**49, 0**	**wparam = 49d = 31hex = escrow information** **lparam = 0d = Return lever activated** **customer wants his money back -> pay out coins inserted = Setpaymentmanager(1,0,2100,0) = 2100 (whole amount will be paid out)**	**0.00 $**
Bill validator disconnected	**2, 3**	**wparam = 2d = 02hex = bill validator status** **lparam = 3d = bill validator not found** **-> stop NRI Payment Manager and restart it to find out which payment unit(s) is/are still available**	**0.00 $**
Bill validator reconnected	**2, 0**	**wparam = 2d = 02hex = bill validator status** **lparam = 0d = bill validator found** **-> stop NRI Payment Manager and restart it to initialize new payment unit**	**0.00 $**
Bill validator cash-box removed	**50, 9**	**wparam = 50d = 32hex = bill validator error** **lparam = 9d = cash-box out of position** **-> disable validator until cash-box has been reinstalled**	**0.00 $**

5 Example code

The example for a code listed below (written in C++) indicates how to load the library (LoadPaymentManagerLib()) using function “Openpaymentmanager”. The code also demonstrates the functions “Closepaymentmanager”, “Startpaymentmanager”, “Stoppaymentmanager”, “Setpaymentmanager” and the function for the message routine “Onpaymentmessage”.
The message routine only has to save the two parameters “wparam” and “lparam” and return the result as fast as possible, as the NRI Payment Manager will only continue to work, if the user replies to the Windows messages. For more details on the Windows message system of the NRI Payment Manager please refer to Chap. 3.

```
////////////////////////////////////////////////////////////////////////////
// connect to the PaymentManager-Dll

#define WM_PAYMENTMESSAGE       (WM_USER+1)

extern "C" {
    typedef int __stdcall nTypST(void);
    typedef int __stdcall nTypI(HWND,int,int,int,int);
    typedef int __stdcall nTypII(int,int,int,int);
    static HINSTANCE vendlib = NULL;
    static nTypST *openpaymentmanager = NULL;
    static nTypI *startpaymentmanager = NULL;
    static nTypST *stoppaymentmanager = NULL;
    static nTypST *closepaymentmanager = NULL;
    static nTypII *setpaymentmanager = NULL;
}

static BOOL LoadVendLib()
{
    if (vendlib) return TRUE;
    vendlib = LoadLibrary("PaymentManager.dll");
    if (vendlib)
    {
        openpaymentmanager = (nTypST*) GetProcAddress(vendlib,"openpaymentmanager");
        startpaymentmanager = (nTypI*) GetProcAddress(vendlib,"startpaymentmanager");
        stoppaymentmanager = (nTypST*) GetProcAddress(vendlib,"stoppaymentmanager");
        closepaymentmanager = (nTypST*) GetProcAddress(vendlib,"closepaymentmanager");
        setpaymentmanager = (nTypII*) GetProcAddress(vendlib,"setpaymentmanager");
        if (!openpaymentmanager ||
            !startpaymentmanager ||
            !stoppaymentmanager ||
            !closepaymentmanager ||
            !setpaymentmanager)
        {
            // Library not found
            openpaymentmanager = NULL;
            startpaymentmanager = NULL;
            stoppaymentmanager = NULL;
            closepaymentmanager = NULL;
            setpaymentmanager = NULL;
            vendlib = NULL;
            return FALSE;
        }
        return TRUE;
    }
    else
    {
        return FALSE;
    }
}

int PM_Open()
{
    int iRtn = 0x1000;                      // dll not found
    if (LoadVendLib()) iRtn = (*openpaymentmanager) ();
    return iRtn;
}

int PM_Close()
{
    if (vendlib)
        return (*closepaymentmanager) ();
    else
        return FALSE;
}

int PM_Start(HWND hWnd, int devices, int messageConfig, int protocol)
{
    if (vendlib)
        return (*startpaymentmanager) (hWnd, WM_PAYMENTMESSAGE, devices, messageConfig,
protocol);
    else
        return -1;
}

int PM_Stop()
{
    if (vendlib)
        return (*stoppaymentmanager) ();
    else
        return -1;
}

int PM_Set(int command, int selection, int info1, int info2)
{
    if (vendlib)
        return (*setpaymentmanager) (command, selection, info1, info2);
    else
        return -1;
}
```

6 Terminology

Amount:	Certain amount of money that, e.g., has to be paid out. Just like the denomination the amount must be an integer including two decimal places as well.
Cash collector:	Units that collect the accepted money, e.g. cash-box, change tubes of coin changers, hoppers.
Denomination:	Nominal value of a coin or banknote. The denomination must be an integer including two decimal places. Examples: $ 0.25 = 25, $ 1.00 = 100, $ 50.00 = 5000, etc.
Denomination ID:	The ID of the denomination depends on the number of coins and bills programmed in the payment units connected. The denominations are processed in ascending order. Example: The NRI Payment Manager addresses a coin changer and a hopper with $ 0.05, $ 0.10 and $ 0.25 coins: denomination 5 = ID 0, denomination 10 = ID 1, denomination 25 = ID 2 (ID 3 = denomination 0, because it does not exist).
Escrow:	Intermediate cash-box for coins/bills which may be returned if the machine's return button is pressed. If the return button is not pressed and the customer clinches the deal, the coins/bills will be accepted and directed into the cash-box or a payout unit (tubes, hoppers).
Payment unit:	Device that is installed in a machine in order to validate, accept, collect or pay out means of money (coin validators/changers, hoppers, bill validators, cashless systems, etc.).

7 Quick reference

openpaymentmanager(void)

openpaymentmanagerex(int port)

port = communication ports to search for devices

	USB only = 0x10000
	COM1 only = 0x00101
	COM1..15 = 0x0010F
return code:	0x0000 = ok, no error
	0x1000 = no USB Dll (NRIHIDAPI.DLL) found

startpaymentmanager(HWND wid, int mid, int devices, int protocol, int message)

wid:	ID of window that receives Payment Manager messages
mid:	ID for messages of Payment Manager
devices:	0x0000 = search for all
	0x0001 = search for coin mech
	0x0002 = search for bill validator
	0x0004 = search for cashless
	0x0008 = search for hoppers defined in paymentmanager.ini
protocol:	0x0000 = search for all
	0x0001 = search for MDB/S1
	0x0002 = serach for ccTalk
	0x0004 = serach for ccNet
message:	not used yet (set to 0)
result code	0x0001 = coinmech found
(ok)	0x0002 = bill validator found
	0x0004 = cashless system found
	0x0008 = hopper(s) found
	0x0010 = escrow(s) found
	0x0020 = display found
(error)	0x2000 = Payment Manager was not opened
	0x2001 = unknown protocol selected
	0x2002 = error in mdb/usb adapter communication
	0x2003 = error in mdb/usb adapter communication
	0x2004 = no communication with payment units
	0x2005 = Payment Manager is already running
	0x2006 = Payment Manager could not start thread

setpaymentmanager(int cmd, int info1, int info2, int info3)

cmd	info1	info2	info3	result	
0	0	-	-	A	INHIBIT: enable all cash items
0	1	-	-	A	INHIBIT: disable all cash items
0	2	val	-	A	INHIBIT: enable cash item with specified value
0	3	val	-	A	INHIBIT: disable cash item with specified value
0	4	id	-	A	INHIBIT: enable cash item with specified ID
0	5	id	-	A	INHIBIT: disable cash item with specified ID
1	0	val	-	value paid out	PAYOUT: payout specified value (device selected autom.)
1	1	val	dev	value paid out	PAYOUT: payout specified value using specified device
1	2	-	-	value	PAYOUT: get lowest payout value possible
1	3	-	-	value	PAYOUT: get highest payout value possible
1	4	dev	idx	value	PAYOUT: get pyout item(s) of specified device idx = item no. 0..n (device with 2 or more items)
1	5	val	dev	0	PAYOUT: asynchronous
1	6	dev	0	bitmask	PAYOUT: get empty/full status of specified device bit7..0: bit 5 = high level sensor supported bit 4 = low level sensor supported bit 1 = high level reached (dev. full) bit 0 = low level underrun (dev. empty)
1	7	dev	en	A	PAYOUT: enable/disable device for automatic payout, en = enable (1 = enable, 0 = disable), after start-up all devices are enabled
1	8	val	num	A	PAYOUT: set number of bills to be payed out from bill-to-bill (e.g. val = 500, num = 2 for payout of two 5-euro banknotes)
1	8	0	0	A	PAYOUT: pay out bills from bill-to-bill unit set with previous command (first set value for each denomination and then start payout
2	0	id	-	value	STATUS: get value of specified cash item
2	1	id	-	device	STATUS: get device the cash item is programmed in
2	2	id	-	collector	STATUS: get collector the cash item is collected in
2	3	id	-	availability	STATUS: get inhibition state of cash item
2	4	id	-	tube counter	STATUS: get no. of coins in tube/bills in cassette
2	5	id	-	sorter path	STATUS: get cash target/sorter path
3	0	0	0	A	ESCROW: disable escrow for all bills
3	0	0	1	A	ESCROW: enable escrow for all bills
3	1	id	0	A	ESCROW: disable escrow specified cash item
3	1	id	1	A	ESCROW: enable escrow specified cash item
3	3	0	1	A	ESCROW: accept bill from escrow
3	3	0	2	A	ESCROW: return bill from escrow
3	3	0	3	A	ESCROW: extend timeout
3	4	dev	dir	A	ESCROW: open escrow, dev = escrow no. (0=all) dir = direction (1=accept, 2=reject)
3	5	0	0	status	ESCROW: get status (opened/closed) of escrows (LSB = escrow 1, a bit set to 1 -> opened, 0 = closed ()

4	0	id	path	0	SORTER: set sorter path
4	1	data	-	0	SORTER: set sorter override
4	2	cassette	value	A	SORTER: set bill-to-bill recycling cassette value
4	3	0	0	status	SORTER: get status of sorter connected additionally
5	0	0	0	[rows][cols]	DISPLAY: get display size, result 0x0110 = 1x16
5	1	mode	ptr	0	DISPLAY: set text mode = display mode (0=solid text) ptr = pointer to text (char string)
6	0	cas	cnt	num	MAINTENANCE: unload "cnt" bills from bill-to-bill recycling cassette number "cas" to drop cassette returns number of bills unloaded

General result codes A:		
	≥ 0	= OK with value returned
	-1	= cash ID not found, device not found, execution error
	-2	= value too small
	-3	= no device attached
	-4	= device error
	-5	= command unknown/not supported
	-6	= Payment Manager is not running

closepaymentmanager(void)

resultcode	0x0000 = ok
	0x0001 = error

MESSAGES

wparam		**lparam (32 bit)**
bit7..4 = Message type	**bit 3..0 = Generating device**	
0 = Status	0 = Payment Manager 1 = Coin Changer 2 = Bill validator 3 = Cashless system 4 = Hopper	0 = Found (during start-up) 1 = Ready 2 = Out of order (hardware problem) 3 = Disconnected (while running) 4 = Tube of coin changer empty 5 = Not found (during start-up) 6 = Reconnected 7 = Bills removed from dispenser 0x0000 001y = Accepted bill routed to cassette no. y (0 = drop cassette, for bill-to-bill only) 0x0000 1xyy = Payed out no. yy of coin ID x (MDB coin changers only) 0x0001 yyyy = Accepted currency code (ccTalk coin acceptance only)
1 = Cash acceptance		Denomination
2 = Cash payout		Denomination
3 = Escrow		0 = Return lever activated > 0 = Denomination routed to escrow
4 = Error		Error code

ERRORS

Code	Description	Coin validator/ changer	Bill validator	Hopper
0	Communication error	X	X	X
1	Coin validator/changer was reset	X		
2	Bill validator was reset		X	
3	Sensor problem	X		
4	Defective motor	X		
5	Jam in tube	X		
6	Jam in coin validator	X	X	
7	Bill validator ROM checksum error		X	
8	Coin validator/changer ROM checksum error	X		
9	Cash-box removed		X	
10	Defective tube sensor	X		
11	Payment unit disabled	X		
12	Validator unplugged	X		
13	Coin jam	X		
14	Sorting error	X		
15	String recognition	X	X	
16	Cash-box full		X	
17	Jam in cash-box		X	
18	Cash-box error		X	
19	Hopper motor blocked			X
20	Hopper empty			X
21	Hopper optics blocked			X
22	Hopper optics error			X
23	Hopper payout blocked			X
24	Bills in dispenser of bill-to-bill unit to be removed			X
25	Tube cassette removed	X		
26	Sorting opened	X		