

NQuire Application Programming Manual

Yuansheng Liu

Revision 1.0

2016-01-11

Table of Contents

1. Preview	1
2. About The Application Manual	2
3. Working Mode	3
3.1. Server-Client Mode	3
3.2. Offline/Local database mode	3
3.2.1. How to build offline database	4
3.2.2. How to pack offline database	5
3.2.3. How to import offline database	6
4. Discovery Protocol	7
4.1. Introduction	7
4.2. Protocol description	7
5. Server Communication	10
5.1. Protocol introduction	10
5.2. NQuire messages	13
5.3. Event messages	14
5.4. Encryption	16
6. Configuration Guide	17
6.1. Introduction	17
6.2. Format	17
6.3. Comment on some settings	18
6.3.1. Authentication	18
6.4. Default content	19
6.5. Webui Congiguration tool	25
6.5.1. General	25
6.5.2. Examples on some settings	26
7. Control Command	36
7.1. Definition	36
7.2. Supported SG15 compatible escaped codes	36
7.2.1. clear screen	36
7.2.2. set cursor position	36
7.2.3. set pixel position	37
7.2.4. word wrap	37
7.2.5. align string of text	37
7.2.6. nop	38
7.2.7. select font set	38
7.2.8. reboot	38

7.2.9. enable/disable scanning	38
7.2.10. enable/disable backlight	38
7.2.11. sleep/wakeup barcode scanner	39
7.2.12. beep	39
7.2.13. get firmware version	39
7.2.14. get firmware version in SG15 format	39
7.2.15. set GPIO output	39
7.2.16. get GPIO input	40
7.3. Extra escape codes, NQuire specific	41
7.3.1. display a picture	41
7.3.2. display touchscreen button picture	41
7.3.3. show idle message	42
7.3.4. set one-time-timeout and server-event	42
7.3.5. clear text layer	43
7.3.6. read mifare card	43
7.3.7. write mifare card	45
7.3.8. Error Codes	47
7.4. show configuration	47
8. ADD-ONS	48
8.1. Mifare	48
8.1.1. Context	48
8.1.2. Mifare read	48
8.1.3. Mifare write	49
8.1.4. Transaction log-file	49
8.1.5. Messages	50
8.2. Touch Screen	52
8.2.1. Introduction	52
8.2.2. Hardware	52
8.2.3. Constraints	52
8.2.4. Design	53
9. Demo	55
9.1. Applicable devices and environment	55
9.2. How to run the demo	55
9.2.1. More details	57
9.2.2. Special token in bmap.txt	58
9.2.3. How to have NQuire display Unicode characters	59
9.2.4. Some hints	59
9.2.5. A demo for picture display in bmap.txt	59

9.2.6. A demo database for touch screen in bmap.txt	60
9.3. Known problems	60
10. Software Update	61
10.1. Upgrade to latest NQuire firmware using FTP	61
11. Summary	62
12. Tell Us What You Think... ..	63

List of Tables

3.1. exmaple	5
3.2. example for barcodes.csv	5
3.3. example for formats.csv	5

Chapter 1. Preview



NQuire is a self-service terminal which is multi-functional and flexibly configurable. NQuire supports touchscreen operation with a 4.2-inch monochrome LCD screen. It is flexible to configure communication mode in NQuire like LAN, WIFI and GPRS etc. Customer can choose to use scan engine (1D or 2D) and RFID module as required. NQuire have USB host port to support data upload/download and drive external device via GPIO like controlling gate.

Chapter 2. About The Application Manual

The application manual contains the following chapters:

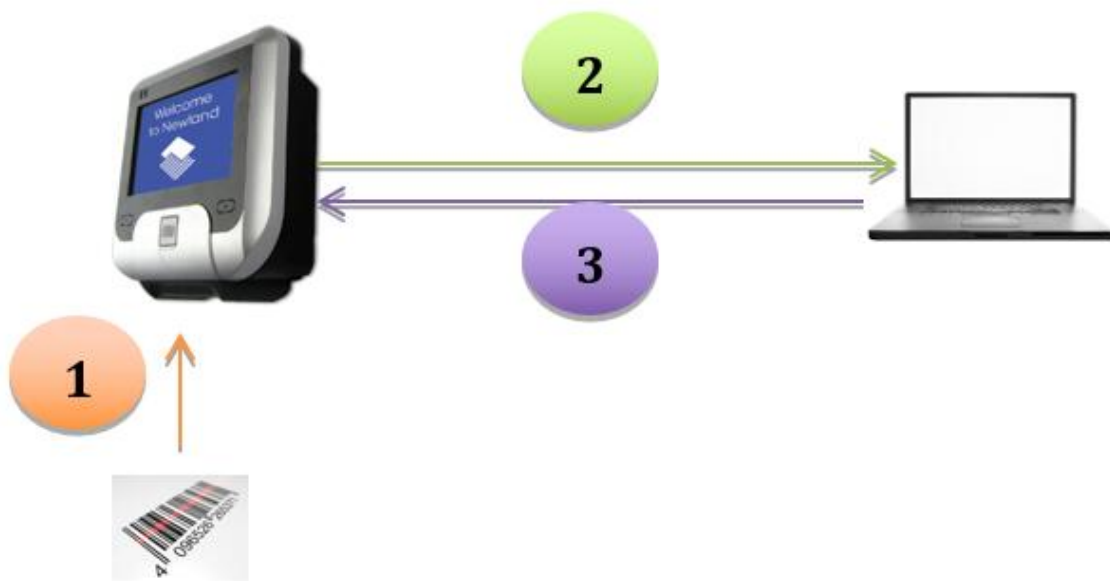
- The chapter "Working Mode" describes operating mode in NQuire
- The chapter "Discovery Protocol" describes how to discover NQuire in a net work.
- The chapter "Server Communication" describes sever communication supported by NQuire.
- The chapter "Configuration Guide" describes configuration prcess in NQuire.
- The chapter "Control Commands Index" describes escape format in NQuire.
- The chapter "ADD-ONS" describes extensive application in NQuire.
- The chapter "Free Demo" describes a demo program developed by Newland in NQuire.
- The chapter "Software Update" describes how to update firmware in NQuire.

Chapter 3. Working Mode

NQuire as a smart terminal, is capable of working configured in different modes. The method to change configuration is depicted in chapter "[Configuration Guide](#)".

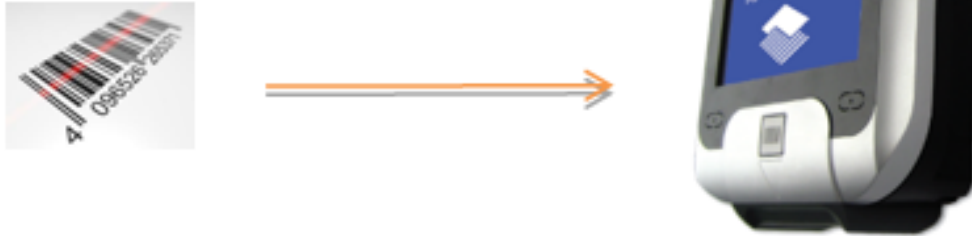
3.1. Server-Client Mode

This is typical working mode on NQuire. In Server- Client mode, NQuire is web client, reports events (eg. scanned barcode,touch-area,etc.) to server via Ethernet. After receiving and processing event from client, sever will send commands to NQuire. Customer also can use the commands to dispaly/update information or control beeper, and even drive GPIO. Please see details in Chapter "[Control Command](#)"



3.2. Offline/Local database mode

In offline/local database mode, NQuire does not send events to server via Internet anymore but instead find the corresponding commands in local database. Customer needs to predefine events(eg.barcode), command list(eg.display price), save as the specified document format(eg.csv) and upload it by ftp or flash disk.



Presently, the supported events for offline/local database: Barcode or RFID reading.



When NQuire works in offline/local database mode, customer needs to modify cit.conf document.

A example is shown in below:

```
/cit/offlinedb/enabled = true
/cit/offlinedb/mode = "server fallback"
/cit/offlinedb/import_busy_msg = "!"
/cit/offlinedb/import_busy_msg_pos = "left-bottom"
/cit/offlinedb/failure = "remove"
/cit/mode = "offline"
```

How to modify cit.conf, please see details in Chapter ["Configuration Guide"](#)

3.2.1. How to build offline database



Offline database consists of two documents: barcodes.csv, formats.csv. The two documents are both .csv format, capable of being exported from Excel or generated manually. Barcodes.csv have two column at least, and user can define multiple columns conveniently.

Table 3.1. exmaple

Name of Column 1	Name of Column 2	Name of Column 3
Cell in column 1, row 1	Cell in column 2, row 1	Cell in column 3, row 1
Cell in column 1, row 2	Cell in column 2, row 2	Cell in column 3, row 2



Name of Column 1 must be named as 'barcode', Name of Column 2 must be named as 'format', Name of Column 3, as add-one can be user-defined(eg. barcode type).

These examples below for barcodes.csv and formats.csv: when 'Newland' is scanned, the barcode defined by format1 shall be shown on screen.

Table 3.2. example for barcodes.csv

barcode	format	code
Newland	format1	Code 128
China	format2	QR Code

Table 3.3. example for formats.csv

format	response
format1	\x1b\x42\x30\x1b\x25\x1b\x2e \x30\${barcode}\x03
format2	\x1b\x42\x30\x1b\x25\x1b\x2e \x30\${barcode}\x03

3.2.2. How to pack offline database

Offline database must be packed as .zip format, and filename format must be "offlinedb-<user define>-<md5checksum>.zip". "<user define>" can be user-defined or default. "<md5checksum>" are 32 bit characters, represents hash value of MD5 for this document.



A freeware to allow user to calculate MD5 hash or checksum¹

A way to pack offline db can be referenced below:

¹ <http://www.winmd5.com>

```
$ cat Makefile
TMPF:=_tmp.zip
CALSUM:=md5sum -b $(TMPF) | cut -b -32

all: zip sum

zip:
rm -f $(TMPF)
zip $(TMPF) barcodes.csv formats.csv
```

generated exmaples:

offlinedb-newland20111020-8b38a1aa2d5ab1dee7a5befe1fbea3cb.zip

offlinedb-8b38a1aa2d5ab1dee7a5befe1fbea3cb.zip

3.2.3. How to import offline database

1. Configure CIT as offline database (see chapter "[Configuration Guide](#)").
2. Upload the packed offline database to ftp.
3. Cit applies offline database automatically.

Chapter 4. Discovery Protocol

4.1. Introduction

The NQuire discovery protocol can be used to discover all NQuire devices in a network.



Another function is the get information about the available NQuires.

4.2. Protocol description

The protocol is fairly simplistic:

The server has to send an UDP packet to the discovery port (19200) of the NQuire (using broadcasting on 239.255.255.250 or the actual address of the NQuire), containing the following text:

```
.....  
CIT-DISCOVER-REQUEST  
Version: 1  
.....
```

In which:

- Line breaks can be '\n' (0x0d) or '\n\r' (0x0d0a) or '\r\n'

- The first line is a discovery packet identifier. It should contain: `CIT-DISCOVER-REQUEST` (cit stands for 'customer information terminal').
- The second line is the version of the discovery protocol (so this is not the version of the application!)

It begins with the "Version:" string (case sensitive, no space between "Version" and ":"), followed by exactly 1 space character and the version number "1". The protocol is somewhat relaxed: there can be a number of spaces and/or tabs between the ":" and the version number.

- The protocol is case sensitive, so shown messages should be send exactly as shown.

The discovery protocol is extended with an optional line containing: `RESPONSE-TO-SENDER-PORT`. When the NQuire recieves a discovery packet with this option, it will send the response packet to the port from where the discovery packet was send. Otherwise it will be send to port 19200.

The discovery protocol is extended with an optional line containing: `RESPONSE-TO-SENDER-ADDRESS`. When the NQuire recieves a discovery packet with this option, it will send the response packet to the address from where the discovery packet was send. Otherwise it will be broadcasted to 239.255.255.250.

The protocol version is not increased because this addition is of no consequence when not used.

All NQuires receiving the packet and implementing the sent version will respond by broadcasting an UDP packet to `239.255.255.250` port `19200` or, when `RESPONSE-TO-SENDER-PORT` is used, to the sender port of the `CIT-DISCOVER-REQUEST` packet.

The packet will have the following format (BNF):

```
.....
file ::= "CIT-DISCOVER-RESPONSE" EOL <idvalues>
idvalues ::= <idvalue> EOL <idvalues> | <idvalue> EOF
idvalue ::= <identifier> ":" SPACE <value>
SPACE ::= " " | " " SPACE | TAB SPACE
identifier ::= [printable ascii not ':']
.....
```

NQuire discovery protocol will send the following packet header and identifier-value pairs (values are an example):

CIT-DISCOVER-RESPONSE

Device name: Newland NQuire 200
Application version: 2.0
Application build nr: 456
Serial number: EI0123456RW
IP-Address: 192.168.1.226
MAC-ethernet: <wifi>
MAC-wifi: <mac of wifi>
Hardware version: <e.g. 2.0>
Firmware version: <e.g. 2.0.2>
Root file system version: <...>
scanner: <scanner module software version>
mifare-model: <mifare model version number>
touch-pad: <touchpad model info>
micro-sd: <micro sd card type info>

Note that items of optional hardware are skipped when the hardware option is not installed.

The protocol version is not changed because the existing tag-value pairs are unchanged, and the format is unchanged, so unrecognized tag-value pairs can simply be ignored.

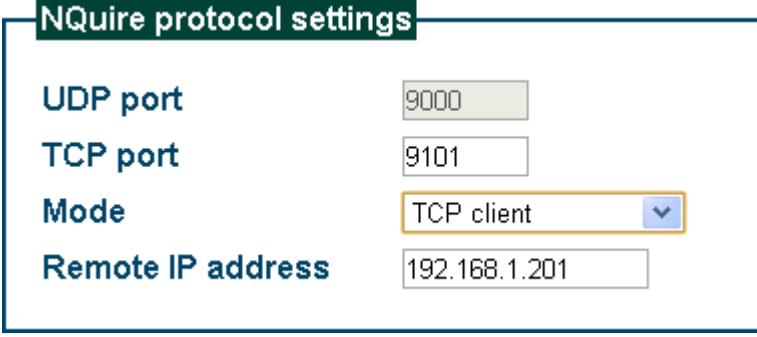
- Note 1: The protocol for a certain version will not change. Any change with a consequence for usage of the existing protocol version will result in an increased version number in the `CIT-DISCOVER-REQUEST` packet.
- Note 2: The rational behind using broadcasting is to be able to discover NQuire configured for a different subnet as the server sending the discovery packet.

Chapter 5. Server Communication

The server communication is done by UDP and/or TCP. The TCP connection can be in server mode or in client mode. In server mode the NQuire will listen to connections on the configured TCP port, whereas in client mode the NQuire will try to make the connection using the configured "remote ip address" and "tcp port".

5.1. Protocol introduction

- TCP client



The screenshot shows a window titled "NQuire protocol settings" with a dark green header. Inside, there are four settings: "UDP port" with a text box containing "9000", "TCP port" with a text box containing "9101", "Mode" with a dropdown menu showing "TCP client" and a blue downward arrow, and "Remote IP address" with a text box containing "192.168.1.201".

The NQuire will try to make a TCP connection to `/cit/remote_ip` and `/cit/tcp_port` continuously. Event messages are sent to the connection server. The NQuire accepts escape code commands from the tcp connection.

Use TCP client, UDP port does not make sense.

NQuire requires handshaking with `192.168.1.201:9101`(TCP server) when starting up and try to keep connection. Scanned barcodes is sent to TCP server directly (no need three-way handshake since connection is established). If failed to establish TCP connection (eg.server crash), NQuire would inform "failure response"(display Please Ask for Assistance on screen)

- TCP client on scan

NQuire protocol settings

UDP port	<input type="text" value="9000"/>
TCP port	<input type="text" value="9101"/>
Mode	<input type="text" value="TCP client on scan"/>
Remote IP address	<input type="text" value="192.168.1.201"/>

Similar to "TCP Client"

The NQuire will try to make a TCP connection to /cit/remote_ip and /cit/tcp_port when there is no connection yet, but only when something is to be send. Then the event message is send to the server. The NQuire accepts escape code commands from the tcp connection (when there is one).

Differently, NQuire establish TCP connection only when transmitting data; while "TCP client" try to keep connection always whatever if demands transmitting data. Note: When NQuire does not establish TCP connection, server cannot send data to NQuire.

- TCP server

NQuire protocol settings

UDP port	<input type="text" value="9000"/>
TCP port	<input type="text" value="9101"/>
Mode	<input type="text" value="TCP server"/>
Remote IP address	<input type="text" value="192.168.1.201"/>

The NQuire will listen to incoming TCP connection on port /cit/tcp_port. Event messages will be send to all open tcp connections. The NQuire accepts escape code commands from TCP connections.

After startup, NQuire cannot send scanned barcode (since she does not know who is available to receive) until a client(e.g.a PC) require a TCP handshake



"TCP server" is not compliant in a situation where NQuire needs "transmitting data" positively but it is compliant where needs "receiving data" passively.

- UDP

NQuire protocol settings

UDP port	<input type="text" value="9000"/>
TCP port	<input type="text" value="9101"/>
Mode	<input type="text" value="UDP"/> ▼
Remote IP address	<input type="text" value="192.168.1.201"/>

events messages are only send using UDP to the configured server (/cit/remote_ip and /cit/udp_port). The NQuire accepts escape code commands from UDP messages.

In this way, set TCP port does not make sense since communication depepnds on UDP only.

Tick UDP, There is no difference for concept between "sever" and "client". You can think NQuire are server and also client.

e.g. UDP port:9000, remote_ip:192.168.1.201

NQuire needs to receive data from 192.168.1.201, and send data to 192.168.1.200:9000 at the same time.



UDP is relatively compliant in a situation where NQuire and PC both needs "Transmitting data" positively.

- client

NQuire protocol settings

UDP port	<input type="text" value="9000"/>
TCP port	<input type="text" value="9101"/>
Mode	<input type="text" value="client"/> ▼
Remote IP address	<input type="text" value="192.168.1.201"/>

The NQuire will try to make a TCP connection to /cit/remote_ip and /cit/tcp_port continuously. Event messages are send to the connection server and using UDP to the configured server (/cit/remote_ip and /cit/udp_port). The NQuire accepts escape code commands from TCP connections and UDP messages on port /cit/udp_port.

e.g. e.g. UDP port:9000, TCP port:9101, remote_ip:192.168.1.201

NQuire implement "UDP" and "TCP client" at the same time. After scanning barcode, NQuire will send data to 192.168.1.201:9000(UDP) and 192.168.1.201:9101(TCP).

- server

NQuire protocol settings

UDP port	<input type="text" value="9000"/>
TCP port	<input type="text" value="9101"/>
Mode	<input type="text" value="server"/> ▼
Remote IP address	<input type="text" value="192.168.1.201"/>

The NQuire will listen to incoming TCP connection on port /cit/tcp_port. Event messages will be send to all open tcp connections and using UDP to the configured server (/cit/remote_ip and /cit/udp_port). The NQuire accepts escape code commands from TCP connections and UDP messages on port /cit/udp_port.

NQuire implement "UDP" and "TCP server"

5.2. NQuire messages

There are two types of data traffic comming from the NQuire:

- event messages: This is data generated in reaction to an event on the nquire (e.g. a scanned barcode or a touchscreen keyclick) These messages are sent to all connected clients and possibly the configured server (UDP or TCP).
- return data: This is data which is generated in response of executing an escape command. This is only send back to the client that issued the command.

Depending on the configuration, messages can be tagged:

- barcode event messages are prefixed with the barcode id (config `/dev/scanner/enable_barcode_id` or webui)
- and a free format tag using `/cit/enable_message_tag`, `/cit/message_tag`:

```
# serial number
# mac-address
# current date/time (as of version 2.0)
```

5.3. Event messages

Simply, event messages is information that we can receive from NQuire.

Event messages are sent to each connected host and, like when UDP communication is enabled, to the configured server-ip.port address. This means that a connection which is used to send commands, will also get event messages. In fact, any

Event messages are separated by the configured message separator (/cit/message_separator): LF,CR or CRLF.

Messages are composed as follows (BNF):

```
.....
message ::= [ device_id ] { event_message | control_cmd_return_message }
device_id ::= <depend on /cit/enable_message_tag and /cit/message_tag>
event_message ::= prefix data
control_cmd_return_message ::= data
prefix ::= <see below>
data ::= <message specific>
.....
```

When /cit/enable_message_tag = true , then each message begins with a device identifier specified in /cit/message_tag.

Different event messages can be recognized by their prefix:

Name	Prefix hid	prefix out
Code128	j	#
UCC_EAN-128	u	P
EAN-8	g	FF
EAN-13	d	F
UPC-E	h	E
UPC-A	c	A
Interleaved-2_of_5	e	i
Code39	b	*
Codabar	a	%
Code93	y	c
PDF417	r	r
QR_Code	S	s
Aztec	Z	z
DataMatrix	U	u
Chinese-Sensible	H	h
GS1_Databar	R	R
ISBN	B	B
Code-11	z	Z
2_5-Matrix	v	v
ITF14	q	q
MSI-Plessey	m	m
Plessey	p	n
2_5-Standard	s	o
2_5-Industrial	i	o
USB		U
mifare		MF
timeout		T
touch16		K
gpio		I
warning		W



When configuration option `"/dev/extscanner/raw = true"`, the data as received from the external scanner is send to the server, prefixed with an 'U': no prefix translation is performed!



The data received on the USB is chopped in chunks. The chunks received from the USB are to be seperated by CR or LF.

5.4. Encryption

Traffic from and to the NQuire can be encrypted. The only encryption supported is base64 (not really encryption but it is no longer human readable text). This is configurable using config item: `/cit/message_encryption`

Default operation mode is 'none' in which case all messages are transmitted as they are.

When `/cit/message_encryption = base64` , the following traffic is encoded:

- All event traffic (barcode, mifare, touchscreen, gpio events) from NQuire attached clients is encoded in base64. The base64 lines are separated with the configured message separator (config item: `/cit/message_separator`). These are only for separating the base64 strings. The actual messages will contain the event(s). Each event ends with the configured message separator.
- All escape codes traffic to the NQuire is expected to be encoded the same manner.



discovery packets are NOT encoded.

Chapter 6. Configuration Guide

6.1. Introduction

The NQuire can be configured using a configuration file: `cit.conf`. This file can be maintained using the webui, or manually using a text-editor. When using the text-editor, one should first download the file from the NQuire using ftp. Then the file has to be changed, after which the file can be uploaded using ftp.

The configuration file can also be used to reset the configuration when the NQuire has become unreachable (eg passwords forgotten). Resetting is done by putting a `cit.conf` on a micro-sd card and insert that in the device. During boot, that file will be used instead.



When changing the file, one should adhere to the format of the file.

6.2. Format

The configuration file is a line oriented file.

Each line contains one of:

- an empty line
- a comment line (starting with a '#')
- a configuration item.

A configuration line is formatted as follows:

```
<config-id> '=' <value>
```

or

```
<config-id> '=' "<value>"
```

The config-id exists of a path and a name. The '=' sign can be surrounded by space or tab characters, which will be ignored. When a value has spaces in it, one can use quotes. All string values will be quoted when generated by the NQuire application.

Only escape characters that have any meaning are implemented for string values. These are:

```
\n translates as a new-line character (=x0a)
\" translates as a quote (")
\\ translates as (\)
\xnn translates as the hexadecimal character value equal to nn
```

When the escape character '\' is used before any other character, it is just left as is.

Eg:

```
/dev/mifare/msg/incomplete_scan/text = "Wait for \"beep\" when scanning"
```

Note that using control characters other than '\n' has no meaning for string values such as the welcome message.

Consequently, the webui has no way of dealing with these escapes.

6.3. Comment on some settings

6.3.1. Authentication

When authentication is turned on in the webui, the NQuire will be username and password protected. The same username and password is used for the webui as for ftp access.

E.g.:

```
# Authentication

/dev/auth/enable = true
/dev/auth/username = "piet"
/dev/auth/encrypted = "BwVYu14F3zRDalosNitNU/"
```

The password is not stored unencrypted. It is possible to use a settings barcode or a cit.conf to change the authentication of the device.



the encrypted password has to be generated by the NQuire. This can be done by setting the password using the webui.



enabling authentication (either by magic barcode or by upload of cit.conf) should always be accompanied by an encrypted password.



the username is NOT reset when disabling authentication. This means that enabling authentication with only /dev/auth/enable and /dev/auth/encrypted will work using the previous password!

6.4. Default content

The default content as of application version 2.0 is:

```
# Settings

# CIT settings

/cit/codepage = "ibm852"
/cit/tcp_port = 9101
/cit/udp_port = 9000
/cit/mode = "server"
/cit/remote_ip = "192.168.1.201"
/cit/http_address = "http://"
/cit/disable_scan_beep = false
/cit/programming_mode_timeout = 15
/cit/loglevel = 4
/cit/message_separator = "LF"
/cit/message_encryption = "none"
/cit/enable_message_tag = false
/cit/message_tag = "${serial}:"
/cit/backdoor = false

# Offline database

/cit/offlinedb/enabled = false
/cit/offlinedb/mode = "server fallback"
/cit/offlinedb/import_busy_msg = "${progress}% "
/cit/offlinedb/import_busy_msg_pos = "left-bottom"
/cit/offlinedb/failure = "remove"
```


Messages

Font sizes

/cit/messages/fontsize/small = 18

/cit/messages/fontsize/large = 28

Idle message

/cit/messages/idle/timeout = 3

#

/cit/messages/idle/picture/show = false

/cit/messages/idle/picture/xpos = 0

/cit/messages/idle/picture/ypos = 0

Line 1

/cit/messages/idle/1/text = "Welcome"

/cit/messages/idle/1/xpos = 0

/cit/messages/idle/1/ypos = 10

/cit/messages/idle/1/valign = "top"

/cit/messages/idle/1/halign = "center"

/cit/messages/idle/1/size = "large"

Line 2

/cit/messages/idle/2/text = "Scan your product"

/cit/messages/idle/2/xpos = 0

/cit/messages/idle/2/ypos = 50

/cit/messages/idle/2/valign = "top"

/cit/messages/idle/2/halign = "center"

/cit/messages/idle/2/size = "small"

Line 3

/cit/messages/idle/3/text = "# ↓ ↓ #"

/cit/messages/idle/3/xpos = 0

/cit/messages/idle/3/ypos = 80

/cit/messages/idle/3/valign = "top"

/cit/messages/idle/3/halign = "center"

/cit/messages/idle/3/size = "small"

```
# Error message

/cit/messages/error/timeout = 1

# Line 1

/cit/messages/error/1/text = "Please ask"
/cit/messages/error/1/xpos = 0
/cit/messages/error/1/ypos = 30
/cit/messages/error/1/valign = "top"
/cit/messages/error/1/halign = "center"
/cit/messages/error/1/size = "small"

# Line 2

/cit/messages/error/2/text = "for assistance"
/cit/messages/error/2/xpos = 0
/cit/messages/error/2/ypos = 60
/cit/messages/error/2/valign = "top"
/cit/messages/error/2/halign = "center"
/cit/messages/error/2/size = "small"

# Network settings

/network/interface = "ethernet"
/network/dhcp = false

#

#

/network/tcp_keepalive/use_keepalive = true
/network/tcp_keepalive/time = 60
/network/tcp_keepalive/intvl = 20
/network/tcp_keepalive/probes = 6

# IP settings

/network/ip/address = 192.168.1.200
/network/ip/netmask = 255.255.255.0
/network/ip/gateway = 192.168.1.1
/network/ip/ns1 = 192.168.1.1
/network/ip/ns2 = 192.168.1.1
```

Wlan settings

```
/network/wifi/ssid = "default"  
/network/wifi/keytype = "off"  
/network/wifi/key = "1122334455"
```

GPRS/UMTS settings

```
/network/gprs/pin = 0000  
/network/gprs/username = ""  
/network/gprs/password = ""  
/network/gprs/apn = "internet"  
/network/gprs/number = "99**1#"
```

Device settings

```
/dev/name = "Newland NQuire 200"
```

Authentication

```
/dev/auth/enable = false  
/dev/auth/username = "admin"  
/dev/auth/encrypted = ""
```

Barcode authentication

```
/dev/barcode_auth/enable = false  
/dev/barcode_auth/security_code = "0000"
```

Modem

```
/dev/modem/device = "/dev/ttyS1"  
/dev/modem/baudrate = 9600
```

Scanner

```
/dev/scanner/barcodes = "1D and 2D"  
/dev/scanner/enable_barcode_id = true  
/dev/scanner/prevent_duplicate_scan_timeout = NOP  
/dev/scanner/illumination_led = "Blinking"  
/dev/scanner/default_illumination_leds = "On"  
/dev/scanner/reading_sensitivity = "Low"  
/dev/scanner/aiming_led = "Blinking"  
/dev/scanner/1d_scanning_mode = "Sensor mode"
```

```
/dev/scanner/multi_reading_constraint = "Semi"
/dev/scanner/em1300_pre_init = ""
/dev/scanner/em1300_post_init = ""
/dev/scanner/em2027_pre_init = ""
/dev/scanner/em2027_post_init = ""

# Barcodes

/dev/scanner/enable-disable/UCC_EAN-128 = true
/dev/scanner/enable-disable/EAN-8 = true
/dev/scanner/enable-disable/EAN-13 = true
/dev/scanner/enable-disable/UPC-E = true
/dev/scanner/enable-disable/UPC-A = true
/dev/scanner/enable-disable/Interleaved-2_of_5 = true
/dev/scanner/enable-disable/Code39 = true
/dev/scanner/enable-disable/Codabar = true
/dev/scanner/enable-disable/Code93 = true
/dev/scanner/enable-disable/GS1_Databar = true
/dev/scanner/enable-disable/ISBN = true
/dev/scanner/enable-disable/Code-11 = true
/dev/scanner/enable-disable/2_5-Matrix = true
/dev/scanner/enable-disable/ITF14 = true
/dev/scanner/enable-disable/MSI-Plessey = true
/dev/scanner/enable-disable/Plessey = true
/dev/scanner/enable-disable/2_5-Standard = true
/dev/scanner/enable-disable/2_5-Industrial = true
/dev/scanner/enable-disable/PDF417 = true
/dev/scanner/enable-disable/QR_Code = true
/dev/scanner/enable-disable/Aztec = true
/dev/scanner/enable-disable/DataMatrix = true
/dev/scanner/enable-disable/Chinese-Sensible = true

# External Scanner

/dev/extscanner/device = "/dev/tty0"
/dev/extscanner/raw = false

# Mifare scanner

/dev/mifare/device = "/dev/ttyS2"
/dev/mifare/key_A = "FFFFFFFFFFFF"
```

```
/dev/mifare/relevant_sectors = "0:0,15:2"
/dev/mifare/prevent_duplicate_scan_timeout = 3
/dev/mifare/cardnum_format = "hexadecimal"
/dev/mifare/send_cardnum_only = false
/dev/mifare/sector_data_format = "base 64"
/dev/mifare/sector_data_seperator = "none"
/dev/mifare/suppress_beep = false

# Error messages

# Message for

/dev/mifare/msg/incomplete_scan/text = "Wait for beep\nwhen scanning"

# Message for

/dev/mifare/msg/access_violation/text = "Card access\ndenied"
/dev/mifare/msg/transaction_error_message = "Transaction failed!\n\nThis is
logged."

# Touch keypad

/dev/touch16/device = "/dev/event0"
/dev/touch16/prefix = "K"
/dev/touch16/timeout = 60
/dev/touch16/keyclick = "beep1"
/dev/touch16/beep1 = "03a32"
/dev/touch16/beep2 = "03c100"
/dev/touch16/beep3 = "03g32"
/dev/touch16/minimum_click_delay = 1
/dev/touch16/invert = false
/dev/touch16/send_active_keys_only = true

# GPIO

/dev/gpio/prefix = "I"
/dev/gpio/method = "On read GPIO"
/dev/gpio/poll_delay = 15
/dev/gpio/event_counter = false

# Beeper

/dev/beeper/device = "/dev/beeper"
```

```

/dev/beeper/volume = 4
/dev/beeper/beeptype = 1
/dev/beeper/tune_startup = "o3c16g16"
/dev/beeper/tune_shutdown = "o3g16c16"
/dev/beeper/tune_error = "o3c16o2f8"
/dev/beeper/tune_1 = "o3c32"
/dev/beeper/tune_2 = "o4c32"
/dev/beeper/tune_3 = "o5c32"

# Display

/dev/display/mode = "240x128m"
/dev/display/contrast = 1
/dev/display/contrast_min = 100
/dev/display/contrast_max = 160

# End

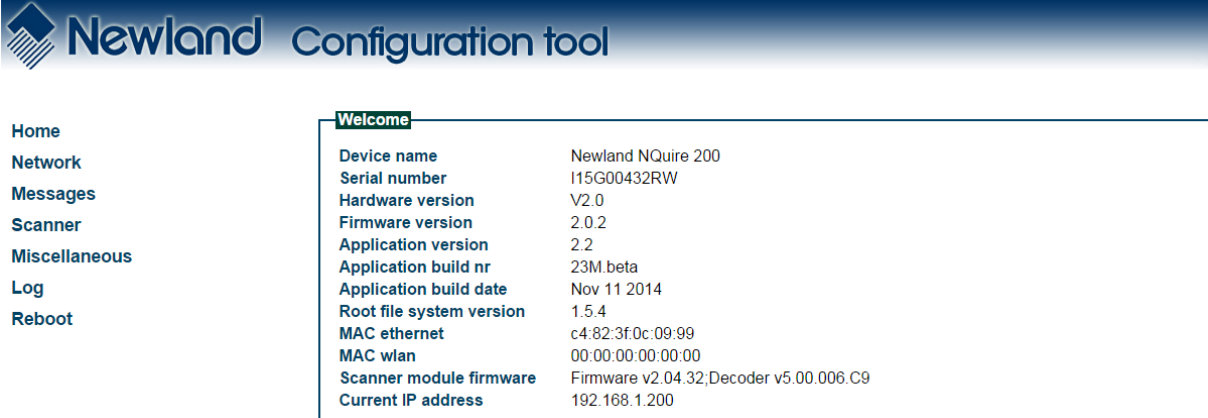
```

6.5. Webui Congiguration tool

6.5.1. General

The NQuire 200/230 uses a internal webserver for configuration. This eliminates Operating System restrictions. You can access the configuration tool by following this process:

1. Open/Start your web browser.
2. Enter the NQuire IP address in the address bar (default 192.168.1.200).
3. The following screen opens:



Welcome	
Device name	Newland NQuire 200
Serial number	I15G00432RW
Hardware version	V2.0
Firmware version	2.0.2
Application version	2.2
Application build nr	23M.beta
Application build date	Nov 11 2014
Root file system version	1.5.4
MAC ethernet	c4:82:3f:0c:09:99
MAC wlan	00:00:00:00:00:00
Scanner module firmware	Firmware v2.04.32;Decoder v5.00.006.C9
Current IP address	192.168.1.200

6.5.2. Examples on some settings

Network settings

When you are using an Ethernet/PoE NQuire and you click on "Network" in the Configuration tool, the following screen opens:

Network interface

Network interface

ethernet ▾

IP Settings

Use DHCP

☒ No
 ☐ Yes

IP address

192.168.1.200

Netmask

255.255.255.0

Gateway

192.168.1.1

Nameserver 1

192.168.1.1

Nameserver 2

192.168.1.1

NQuire protocol settings

Mode

server ▾

UDP port

9000

TCP port

9101

Remote IP address

192.168.1.201

Http address

http://

- IP settings: Use of DHCP (automatic assignment of IP-address to NQuire 200) or not (NQuire has fixed IP-address). In a DHCP-request the NQuire vendor ID is: NQuire200

- NQuire protocol settings:

Define UDP/TCP port;

Define connect mode: server (also UDP active), client (also UDP active), (pure) UDP, TCP server (no UDP), TCP client (no UDP) and TCP client on scan (for non-continuous Ethernet connections such as ISDN). In server mode the NQuire will listen to connections on the configured TCP port, whereas in client mode the NQuire will try to make the connection using the configured "remote ip address" and "tcp port". When there are multiple connections (server mode only), scanned

barcode data will be sent to all connected servers and sent to the configured UDP server:port.

Define remote IP address.

Wireless settings

When you have an Ethernet/WiFi NQuire, two extra boxes appears in the "Network" screen as shown below:

The screenshot shows two sections of a configuration interface. The top section, titled "Network interface", contains a label "Network interface" and a dropdown menu currently set to "wifi". The bottom section, titled "Wifi", contains three fields: "ESSID" with a text input containing "default", "Wireless key type" with a dropdown menu set to "off", and "Wireless key" with a text input containing "1122334455".

- Wireless key type: You can choose between three security levels:

None: No encryption key is needed, the NQuire is, via your wireless router, available to all WiFi enabled devices.

WEP: Entry-level encryption with a wireless key to limit network access.

WPA / WPA2: High-end encryption with a wireless key to limit network access.



It is strongly advised to use a wireless key to avoid third parties to intrude your network. Please ask your administrator what network security level is available in your user environment.

- Wireless key: Type the key which is going to be used to encrypt wireless data communication.



To avoid interference of Access Points outside your own network, please select a channel on your AP which is not used (intensively).

Idle screen settings

When you click on "Messages" in the Configuration tool, the following screen opens:

Idle message						
Text	X Pos	Y Pos	Vert Align	Hor Align	Size	
<input type="text" value="Welcome"/>	<input type="text" value="0"/>	<input type="text" value="10"/>	<div style="border: 1px solid #ccc; padding: 1px;">top ▼</div>	<div style="border: 1px solid #ccc; padding: 1px;">center ▼</div>	<div style="border: 1px solid #ccc; padding: 1px;">large ▼</div>	
<input type="text" value="Scan your product"/>	<input type="text" value="0"/>	<input type="text" value="50"/>	<div style="border: 1px solid #ccc; padding: 1px;">top ▼</div>	<div style="border: 1px solid #ccc; padding: 1px;">center ▼</div>	<div style="border: 1px solid #ccc; padding: 1px;">small ▼</div>	
<input type="text" value="↓↓↓↓"/>	<input type="text" value="0"/>	<input type="text" value="80"/>	<div style="border: 1px solid #ccc; padding: 1px;">top ▼</div>	<div style="border: 1px solid #ccc; padding: 1px;">center ▼</div>	<div style="border: 1px solid #ccc; padding: 1px;">small ▼</div>	
Picture <input type="checkbox"/>	<input type="text" value="0"/>	<input type="text" value="0"/>				

Font sizes	
Font size small	<div style="border: 1px solid #ccc; padding: 1px;">18 ▼</div>
Font size large	<div style="border: 1px solid #ccc; padding: 1px;">28 ▼</div>

Error message					
Text	X Pos	Y Pos	Vert Align	Hor Align	Size
<input type="text" value="Please ask"/>	<input type="text" value="0"/>	<input type="text" value="30"/>	<div style="border: 1px solid #ccc; padding: 1px;">top ▼</div>	<div style="border: 1px solid #ccc; padding: 1px;">center ▼</div>	<div style="border: 1px solid #ccc; padding: 1px;">small ▼</div>
<input type="text" value="for assistance"/>	<input type="text" value="0"/>	<input type="text" value="60"/>	<div style="border: 1px solid #ccc; padding: 1px;">top ▼</div>	<div style="border: 1px solid #ccc; padding: 1px;">center ▼</div>	<div style="border: 1px solid #ccc; padding: 1px;">small ▼</div>

- Idle message: You can type (on three lines) the text which is displayed on the screen at moments nothing is scanned:
 - # X Pos / Y Pos: define the X-and Y positions per pixel on the screen.
 - # Vert Align / Hor Align: Vertical and horizontal alignment options which have system default X and Y screen positions. X positions can only defined when horizontal alignment is set to "left".
 - # Size: Choose between system default large and small size text.
 - # Picture: Instead of or as a background picture in addition to the idle messages you can upload an (animated max. 32 frames and 2 frames per sec.) .gif picture file (2-colour black&white, inverted (black bkgrd/white drawing), max. 240 x 128 pixels, max. 16K size, non-interlaced) through any FTP program. The file name must be renamed to "welcome.gif" before uploading. Please upload to the NQuire "/img" directory. You can now enable the picture.
- Error message: You can type (on two lines) the text which is displayed on the screen when the NQuire receives a timeout from the network (NQuire not connected / offline). Timeout settings can be changed at "Miscellaneous".
- Font sizes: You can select here whether the small used font size should be 18 (default) or 24 pixels high. For instance when you use 18 pixels, you can have up to 25 characters on one line and have 8 lines on a display. For the big font size you can choose between a height of 28 or 32 pixels.



For each setting you want to change and save, click the "Apply settings" button after each change.

Scanner settings

When you click on "Scanner", the following screen opens:

Barcodes

Barcodes

1D only ▾

Constrain multi reading

Semi ▾

Duplicate scan timeout

NOP ▾

Enable barcode ID

☐ No ☒ Yes

UCC/EAN 128

☐ No ☒ Yes

EAN-8

☐ No ☒ Yes

EAN-13

☐ No ☒ Yes

UPC-E

☐ No ☒ Yes

UPC-A

☐ No ☒ Yes

Interleaved 2 of 5

☐ No ☒ Yes

ITF14

☐ No ☒ Yes

Code 39

☐ No ☒ Yes

Codabar

☐ No ☒ Yes

Code 93

☐ No ☒ Yes

GS1 Databar

☐ No ☒ Yes

Code 11

☐ No ☒ Yes

Scanning modes Imager

Red Illumination LEDs

Blinking ▾

Blinking activation sensitivity

Low ▾

Green aiming LED

Blinking ▾

External scanner

Raw data

☒ No ☐ Yes

- Barcodes:
 - # The NQuire reads, by default, only 1D codes. You can select it to read 2D codes as well (only available for NQuire 202 models).
 - # Constrain multi reading:
 - # ON: Same barcode cannot be read twice.
 - # Semi (default): Same barcode can only be read after barcode is removed from scanning field first.

- # OFF: Same barcode can be read twice when in scanning field.
- # Duplicate scan timeout can be set to avoid that e.g. a ticket is scanned twice.
- # The barcode identifiers (as described on page 6) can be enabled or disabled.
- # Depending on the selection made (1D or 1D and 2D), you can select which barcodes you want to enable or disable.
- External scanner:
 - # You can attach an external device to the USB port of the NQuire. When you attach a Newland barcode scanner to it, select "No".
 - # When you attach another device, such as a swipe card reader, external RFID or chipcard reader, please select "Yes". The operating mode of this device must be in USB-HID or USB-KBW mode and should be able to support USB version 1.1. The server will identify this data as it will get the prefix "u" in front of the information read from the external/universal reader.

When you have a RFID module integrated in the Nuire (NQ2xxM1x models) the following option is added to the "Scanner" screen:

Mifare scanner

Access key A	FFFFFFFFFFFF
Sectors to read	0:0,15:2
Cardnum formatting	hexadecimal ▼
Send card number only	<input checked="" type="radio"/> No <input type="radio"/> Yes
Sector data format	base 64 ▼
Sector data separator	none ▼
Suppress scan beep	<input checked="" type="radio"/> No <input type="radio"/> Yes
Duplicate scan timeout	3 ▼
Access violation	Card access\denied
Incomplete scan	Wait for beep\nwhen scanning
Write error message	Transaction failed!\n\nThis is logged.

RFID Scanner settings: When you have a NQuire unit with a RFID scanner integrated, you will be able to configure also the following options:

- Access key: Please type the access key A (no Key B can be inserted) which has been defined for your Mifare tags/cards.
- Sectors to read: Please define which of the 16 sectors of the Mifare tag/ card should be read. Each sector number should be divided by a comma.
- Send card number only: NO sectors will be read, just the cardnumber will be send to the server.

- Sector data format and separator: Manor of encrypting the value in the card and what separator used to separate the different read blocks.
- Duplicate scan timeout: Define how many seconds the RFID reader waits before you can successfully scan the same tag/card with the same information on it (no time restriction when you scan a different tag/card).
- You can change the texts according to your local needs/language.

Miscellaneous settings

When you click on "Miscellaneous", the following screen opens:

Device	
Device name	<input type="text" value="Newland NQuire 200"/>

Offline database	
Enabled	<input checked="" type="radio"/> No <input type="radio"/> Yes
mode	<input type="text" value="server fallback"/>
Busy message	<input type="text" value="\${progress}%"/>
Busy message position	<input type="text" value="left-bottom"/>
On failed import	<input type="text" value="remove"/>

Authentication	
Enable authentication	<input checked="" type="radio"/> No <input type="radio"/> Yes
Username	<input type="text" value="admin"/>
Password	<input type="password" value="****"/>
	<input type="password" value="****"/>

Programming barcode security	
Programming mode timeout	<input type="text" value="15"/>
Enable security code	<input checked="" type="radio"/> No <input type="radio"/> Yes
Barcode programming security code	<input type="text" value="0000"/>

Text and messages	
Idle message timeout	<input type="text" value="3"/>
Error message timeout	<input type="text" value="1"/>
Font codepage	<input type="text" value="ibm852"/>
Scan event separator	<input type="text" value="LF"/>
Message encryption	<input type="text" value="none"/>
Use Custom NQuire identifier	<input checked="" type="radio"/> No <input type="radio"/> Yes
Custom NQuire identifier	<input type="text" value="\${serial}"/>

Interaction

Display contrast	2 ▾
Beeper volume	4 ▾
Beeper type	1 ▾
Disable beep after scan	<input checked="" type="radio"/> No <input type="radio"/> Yes

GPIO

Server message prefix	<input type="text"/>
Append event counter	<input checked="" type="radio"/> No <input type="radio"/> Yes
Method	On read GPIO ▾
Poll speed (seconds)	15 ▾

- Device name: Type a random name used for your own administration.
- Authentication: You can choose whether or not you want a password protection to access the NQuire configuration tool via a username and password.
- Programming barcode security:
 - # Programming mode timeout: The time before returning to idle state when no programming barcode is scanned in seconds.
 - # Security code: You can also program the NQuire with a so-called 2D batch code (only for 202 models, see Appendix) . As you don't not want to allow just anyone to be able to program the unit with barcodes, you can set a security code out of which you can create a barcode to enable programming.
- Text and messages:
 - # Idle message timeout: the period of time before the idle message is displayed again after a scan in seconds.
 - # Error message timeout: the period of time the NQuire device waits for a response from the host pc/server in seconds. When this timeout is exceeded, the error message will be displayed for 5 seconds.
 - # Font codepage: Choose either UTF-8 (universal fontset which supports most used language fonts) or one of the following codepages:

Codepage	Description
851	DOS Greek
852	"Multilingual" West European Latin-1

Codepage	Description
866	Cyrillic DOS codepage
874	Thai
1250	Central and East European Latin
1251	Cyrillic
1252	West European Latin-2
1253	Greek
1254	Turkish
1257	Baltic

The scan message separator (sent after scan) can be set to LF, CR or both.

- When you have an NQuire with touch screen (Model NQuire 23x), one extra box appears in the "Miscellaneous" screen as shown below:

Touch screen

Server message prefix

Touch keyboard timeout [seconds]

Touch key click

Invert button on click ☒ No ☐ Yes

Minimum time between click

Only send active key events to server ☐ No ☒ Yes

- Server message prefix: You can define a prefix so the database identifies the touch "key" similar to identifying a barcode.
- Touch keyboard timeout: The time the keyboard/button is shown on the screen before returning to idle state.
- Touch key click: You can choose between 3 sounds in the event a key is touched.
- Invert button click: When selected "yes" and a button is touched, it automatically inverts without uploading "inverted images" yourself.
- Minimum time between click: Define how many seconds the touch screen waits before you can touch the same key again. Useful to prevent "double clicks".
- Only send active key events to server: When selected "No" every touch is registered on the network, even when no "button" is shown.

- When you click on "Log", the following screen opens:

warning info will be shown in red font.

System log		
1	wrn main	No config file found. Using defaults.
2	inf config	Saved configuration database.
3	wrn main	Detected problematic mac-address 00:05:f4:11:22:33 for eth0
4	inf main	Serial = 43(dec) = 2b(hex)
5	inf main	Changing mac address of eth0 to 00:05:f4:00:00:2b
6	inf main	Ethernet mac 00:05:f4:00:00:2b
7	inf main	Wifi mac 00:00:00:00:00:00
8	inf main	Device serial nr: I43E00220M1RW
9	inf main	Root filesystem version: 1.5.4
10	inf main	Firmware version: 2.0.2
11	inf main	Application version: 2.2, build Elgin v1.0, Jun 30 2015
12	inf main	Open all peripherals
13	inf gpio	gpio opened
14	inf main	Turn on peripheral power
15	inf network	Configuring interfaces
16	inf scan_rf	Mifare hardware model V1 found on device /dev/ttyS2 (fd=12)
17	inf touch16	HW found: AT42QT2160 Touch Sense Keyboard CIT200 version
18	inf webserver	HTTP server listening on port "80"
19	inf network	Configuring interfaces
20	inf network	Bringing up network ethernet interface
21	inf main	Offline server not initialized because there is no mmc memory
22	inf gpio	gpio disabled
23	inf cit	Restoring anonymous ftp-user
24	inf cit	Using font file "arial.ttf"
25	inf main	Start watchdog
26	inf cit	Starting CIT server
27	inf cit	Listening on UDP port 9000
28	inf cit	Listening on TCP port 9101
29	inf main	Starting main event loop
30	inf scanner	Configuring scanner
31	inf scanner	Getting scanner module info
32	inf scanner	Restoring defaults
33	inf scanner	Making basic settings
34	inf scanner	Setting reading constraint and 2d/1d
35	inf scanner	Successfully detected and configured scanner
36	inf network	Configured network successfully: 192.168.1.200
37	inf main	Starting discovery service
38	inf network	Network is up

- Also, logging level can be changed to 'event'(default is info). Use event logging for debugging your escape code messages.

Log settings

Logging level

event ▼

- Reboot Shortcut

When you click on "Reboot", the following screen opens:

Device

Click the button below to reboot the device:

Click the button below to reset factory default settings and reboot the device:

Chapter 7. Control Command

7.1. Definition

NQuire adopts escape codes as control command.

Escape characters are part of the syntax for many programming languages, data formats, and communication protocols. For a given alphabet an escape character's purpose is to start character sequences (so named escape sequences), which have to be interpreted differently from the same characters occurring without the prefixed escape character. An escape character may not have its own meaning, so all escape sequences are of two or more characters.

There are usually two functions of escape sequences. The first is to encode a syntactic entity, such as device commands or special data, which cannot be directly represented by the alphabet. The second use, referred to as character quoting, is to represent characters, which cannot be typed in current context, or would have an undesired interpretation. In the latter case an escape sequence is a digraph consisting of an escape character itself and a "quoted" character.

7.2. Supported SG15 compatible escaped codes

7.2.1. clear screen

.....
`\x1b\x24`
.....

or

.....
`\x1b\x25`
.....

Also see '[clear text layer](#)'.

7.2.2. set cursor position

.....
`\x1b\x27xy`
.....

The x and y values start at 0x30 representing coordinate 0. 0,0 represents resp. left,top.

The valid x and y range is determined by the size of the currently used font.

7.2.3. set pixel position

```
\x1b\x2cxy
```

The x and y values start at 0x30 representing coordinate 0. 0,0 represents resp. left,top.

The valid range is determined by the size of the display (for the NQuire: 240x128).

7.2.4. word wrap

```
\x1b\x2d\x30 word wrap off (default)
\x1b\x2d\x31 word wrap on
```

When turned on, text will be wrapped to the next line instead of being cut off.

7.2.5. align string of text

```
\x1b\x2e<align><text>\x03
```

The align parameter can be one of:

```
\x30 = left top
\x31 = center top
\x32 = right top
\x33 = left middle
\x34 = center middle
\x35 = right middle
\x36 = left bottom
\x37 = center bottom
\x38 = right bottom
\x39 = left (vertical position is not changed)
\x3a = center (vertical position is not changed)
\x3b = right (vertical position is not changed)
\x3c = top (horizontal position is not changed)
\x3d = middle (horizontal position is not changed)
\x3e = bottom (horizontal position is not changed)
```

E.g. "blabla" in left top:

```
\x1b\x2e\x30blabla\x03
```

7.2.6. nop

.....
\x1b\x40
.....

7.2.7. select font set

.....
\x1b\x42<param>
.....

Param can be one of:

.....
\x30 = small
\x31 = large
\x32 .. 0x40 = 6px, 12px, 18px ...
.....

E.g. Set font to the configured 'large' value:

.....
\x1b\x42\x31
.....

7.2.8. reboot

.....
\x1b\x5a
.....

7.2.9. enable/disable scanning

.....
\x1b\x5b<param>
.....

param can be one of:

.....
\x30 = disable
\x31 = enable
.....

7.2.10. enable/disable backlight

.....
\x1b\x5c<param>
.....

Param can be one of:

.....
\x30 = off
\x31 = on
.....

7.2.11. sleep/wakeup barcode scanner

.....
\x1b\x5d<param>
.....

Param can be 1 of:

.....
\x30 = sleep
\x31 = wakeup
.....

7.2.12. beep

.....
\x1b\x5e
.....

7.2.13. get firmware version

.....
\x1b\x5f
.....

This is the NQuire formatted firmware version for the application only. The version is returned on the established connection only (or returned to sender address.port in case of udp communication).

Format is:

.....
<major>.<minor>[.<patch>]
.....

7.2.14. get firmware version in SG15 format

.....
\x1b\x60
.....

7.2.15. set GPIO output

.....
\x1b\x7e<param1><param2>
.....

Parameter definition:

```
Param1: \x30 = OUT1
        \x31 = OUT2
```

```
Param2: \x30 = low
        \x31 = high
```

E.g. setting OUT2 to low:

```
\x1b\x7e\x31\x30
```

7.2.16. get GPIO input

```
\x1b\x7f<param>
```

Param can be 1 of:

```
\x30 = IN1
\x31 = IN2
```

E.g. requesting the state of IN1:

```
\x1b\x7f\x30
```

This send the value back using the following format:

```
<prefix><pin><value>
```

E.g. with the default prefix for IN1, value high:

```
I01
```

It is possible to add a round-robin event counter (modulo 2^{16}) to the GPI event using configuration setting (application version 1.7 and higher):

```
/dev/gpio/event_counter = true
```

Each GPI has a separate counter. By using this counter it is possible to detect missed events and distinguish between change events and 'poll' updates.

E.g. The following shows a missed (high) event between #38 and #40, and a change event #41 and #42. This means that the second #42 event is a polled status-update.

```
I00 38
I00 40
I01 41
I00 42
I00 42
```

7.3. Extra escape codes, NQuire specific

7.3.1. display a picture

Display a picture on the current pixel position. Just specify the filename, close with \x03.

```
\x1b\xfdfilename.gif\x03
```

7.3.2. display touchscreen button picture

The shown picture is related to 1 or more touchscreen buttons.

```
\x1b\xfd2<released.gif>\x0d<pressed.gif>\x0d<position by key-id><coupled to
key-id>\n\x03
```

The meaning of the parameters is explained below:

release.gif = filename of the gif image when the button is not 'pressed'

pressed.gif = filename of the gif image when the button is 'pressed' (NOT IMPLEMENTED)

position by key-id = specify position of left top of image

coupled to key-id = specify to which keys the image is related

When "pressed.gif" is empty, the image of name-released will be inverted when pressed (NOT IMPLEMENTED).

The names of the images should not be too long and not contain spaces. Together they can have $64-16-3=45$ characters

Touch screen position layout:

```
.....  
0 1 2 3  
4 5 6 7  
8 9 a b  
c d e f  
.....
```

E.g. display image on position 5, and associate to key 5

```
.....  
\x1b\xfd21.gif\x0d\x0d55\x03  
.....
```

E.g. display image on position 6, and associate to key 6 and a

```
.....  
\x1b\xfd21.gif\x0d\x0d66a\x03  
.....
```

A [example](#) to show touchscreen in NQuireSvrDemo

7.3.3. show idle message

```
.....  
\x1b\xfd3  
.....
```

This clears the display and shows immediately the idle message (without error message timeout!)

7.3.4. set one-time-timeout and server-event

This command can be used to change the next idle or error message timeout. Additionally, it will send an event when a delay-time is passed.

```
.....  
\x1b\xfd4<type><delay><timeout><tag>\x03  
.....
```

In which:

```
type ::= "I" or "E"  
delay ::= \x31 .. \x7f  
timeout ::= \x31 .. \x7f  
tag ::= some ascii text unique identifying this timeout message  
.....
```

For example:

```
\e\x1bI62takky\x03
```

Would give an event to the server after 6 seconds, then, when the server did not respond with any message within 2 second, the idle display is shown.

The event would be formatted as follows:

```
event ::= 'T' <act><tag>
```

In which:

act ::= 'T' or 'Q' # resp: 'T'=normal timeout or 'Q'=timeout quit (e.g. due to scanning a "enter programming mode" barcode)

E.g.:

TTtakky



no message is send when the timeout is interrupted due to a clear-screen command.

7.3.5. clear text layer

```
\x1b\x1b5
```

7.3.6. read mifare card

```
\x1b\x1b8<cardnum>,<keyA>:{<sector><block><format>}n\x03
```

In which:

cardnum ::= <nibble>8

keyA ::= <hex nibble>12

sector ::= 0x30 .. 0x3f

block ::= '0' or '1' or '2' or '-'

format ::= 'B' or 'H'

Explanation:

cardnum

the cardnum of the card to be written. The transaction is refused when it does not match the current card.

keyA

This is the key that will be used to access the mifare card.

sector

One byte specifying the sector to be written: values 0x30 .. 0x3f respectively representing sector 0 to 15

block

this is the block to be read. The whole sector will be read when using '-'

format

The required format of the returned data.

'B'

The read data is returned as binary (just as it is on the card).

'H'

The read data is formatted as hex-nibbles.

For example, the command for reading sector 1 block 2 formatted as binary, all blocks of sector 4 formatted in hex, and sector 15 block 0 also formatted in hex, using access keyA = "FFFFFFFFFFFFFF" from card with cardnum 76262fa5 looks like:

```
\x1b\xfa876262fa5,FFFFFFFFFFFFFF:\x31\x32B\x34-H\x3f\30H\x03
```

Of course, this will only succeeds when the correct card is presented to the reader of which the specified sectors can be read with the specified access key.

On success, the data is returned. The data of each block (or whole sector) read concatenated, possibly separated by the configured separator character (/dev/mifare/sector_data_seperator). The default separator is 'none'.

```
response-message ::= { ACK <data> {<sep> <data>}* } | { NAK <error code> }
```

```
sep ::= "" | <SPACE> | <TAB> | ",", | ":" | ";"
```

For example, the above could return something like (assuming all blocks are filled with "dirk has access!" and /dev/mifare/sector_data_seperator = "comma"):

```
<ACK>Dirk has  
access!,4469726b20686173206163636573735c4469726b20686173206163636573735c4469726b206861
```



the message separator depends on configuration item /cit/message_seperator (in this case "CR").



when issuing multiple write commands in a row, one should disable mifare-card-detection.

This also shows the limited use of binary-formatting: when the read data contains `,' or <CR> characters it will obfuscate the return string.

7.3.7. write mifare card

By using this command it is possible to write 1 or more blocks of data to a mifare card in 1 transaction.

The transaction id is logged together with the result for each command within the transaction.

Half-way broken transactions are not reverted (it is impossible to do that reliable with a mifare-card).

Instead, the return code and logging provides a way to determine were a transaction went wrong.

```
\x1b\x1b<cardnum>,<transaction-id>:{<cmd>}+\x03
```

In which:

```
cardnum ::= <nibble>8  
transaction-id ::= <nibble>1-8  
cmd ::= {"K" <keyA>} or {"W" <sector> <block> <format+data>}  
keyA ::= <hex nibble>12 sector ::= 0x30 .. 0x3f  
block ::= '0' or '1' or '2' format+data ::= {'B' <byte>16} or {'H'  
  <nibble>32}
```

Explanation:

cardnum

the cardnumber of the card to be written. The transaction is refused when it does not match the current card.

transaction-id

some id (hex) by which the server can identify this write. It should be max 8 hex nibbles long. It is used for transaction logging.

cmd

There can be more than one command. Commands are identified by a prefix: `K', `W', `I' or `D' respectively 'set-key', 'write block', 'increment block' or 'decrement block'. Only 'set key' and 'write block' are implemented.

keyA

This is the key that will be used to access the mifare card.

sector

One byte specifying the sector to be written from values 0x30 to 0x3f respectively representing sector 0 to 15 block.

format+data

The data to be written can be formatted as binary (`B') or as hex-values (`H'). Exact 16 data-bytes are expected when format==`B'. Exact 32 nibbles are expected when format==`H'

For example, transaction '1a2b3c' writing Dirk has access! (note: 16 bytes long!) to block 2 sector 3 of card 0x76262fa5, with access key A = "FFFFFFFFFFFFFF", looks like:

```
\x1b\xfa5,1a2b3c:KFFFFFFFFFFFFFFW\x33\x32BDirk has access!\x03
or
\x1b\xfa5,1a2b3c:KFFFFFFFFFFFFFFW
\x33\x32H4469726b20686173206163636573735c\x03
```

Of course, this will only succeed when a card is presented to the reader of which the specified sectors can be written with the presented access key(s).

Also note that writing in binary-mode is limited to data without the configured EOL control character(s).

In response, the result is returned. On success a `0' character is returned. In case of an error, the result of all sequential commands is returned.

That way, it is possible to see where a transaction went wrong. This is formatted as: response-message ::= { ACK "0" } | { NAK {<cmd><id>}+ }

7.3.8. Error Codes

Error codes are used in return and event messages. They are used to indicate errors and warnings.

```
.....
'0' OK
'1' Error (undefined)
'2' Error: Mifare card unavailable
'3' Error: Mifare Card access denied (incorrect key)
'4' Error: invalid message format
'5' Error: Mifare invalid card (mifare cardnum mismatch)
'6' Error: file system full (no space left to log).
'7' Error: Mifare format error in cardnum or transaction-id
'8' Warning: big-log-file detected (this can slow down the system).
Please rotate and cleanup.
.....
```

7.4. show configuration

```
.....
\x1b\xfe
.....
```

The configuration is shown on the display (e.g. serial number, mac-address, current ip, used interface, etc).

Chapter 8. ADD-ONS

8.1. Mifare

This section describes how the NQuire application handles mifare.

8.1.1. Context

A mifare module is an NQuire hardware option. The current module only supports reading and writing with key A.

The NQuire can be used to read and write mifare classic cards.

8.1.2. Mifare read

There are two ways to read a mifare card using the NQuire:

- off-line: the access key and which sectors should be read are stored (configured) in the NQuire;
- on-line: the access key and sectors to be read are transmitted to the NQuire;

Off-line

When a card is presented to the NQuire, the specified sectors are read using the preconfigured access key. The retrieved data is send to the connected server(s).

On-line

When a card is presented, only the card-num is send to the (connected) server. The server than decides what to do, eg:

- read data from the card
- display a message
- activate the touchscreen

This way it is possible to:

- support multiple keys for reading different sectors
- interact with the user using the touchscreen in-between presenting the card and reading (or writing) the card (eg in case of writing: ask for confirmation).



Do not use off-line mode when mifare keys are to be secret because keys, data and event messages are transmitted unencrypted using tcp and/or udp. It 'is' possible to transmit the message encoded in base64 making them unreadable for humans. However, base64 is easily recognized and decoded.

8.1.3. Mifare write

A mifare write is only implemented on-line. Off course, it is only possible to use mifare write when a mifare card is presented to the NQuire.

The server is notified of presenting a mifare card using the configured mifare read event message.



It is possible to disable to mifare-read-success-beep so there is no 'ok' beep after the read message. That would be confusing because the transacting would not be complete after the beep: a write still has to be done.

On-line

- The server sends a message to NQuire to write data in a certain block.
- When the write succeeds or fails an event message is generated with the result.
- The result is also logged in a persistent log-file (see "Transaction log-file")

8.1.4. Transaction log-file

All mifare write transactions are logged in a logfile:

log/mifare.log is accessible via ftp.

Logfiles are shifted upon the escape command: "shift mifare-log". The logfile will be named as "mifare-<shift id>.log".

The shift id is specified by the "shift mifare transaction log" escape command.

The server can get a "log-file too big" message indicating that the mifare transaction log is too big. This can influence performance. As a response, the server should initiate a "shift mifare log", download the log and remove the log.



When this is not done, it is possible that transactions will fail due to insufficient space to log transactions. The space to log files is also used by uploaded gif-images.

Log file format

Bnf format:

```
line := <transaction-id> {<command> <result>}+
command := "W" | "I" | "D"
result := "
```

8.1.5. Messages

escape commands (server to NQuire)

command	defined as
\x1b \xf8<cardnum>KaaaaaaaaaaaaR{<sector><block>}+ \x03	read mifare card
\x1b\xfb<cardnum><transaction id>KaaaaaaaaaaaaaW<sector><block><format><data> \x03	write data
\x1b\xfb<cardnum><transaction id>KaaaaaaaaaaaaaI<sector><block> \x03	increment block
\x1b\xfb<cardnum><transaction id>KaaaaaaaaaaaaaD<sector><block> \x03	decrement block
\x1b\xfbhhhhhhhh\x03	shift transaction log

More commands see: [read mifare card](#)

Events (NQuire to server)

Event	defined as
MF<cardnum>[<data>]	Mifare data read. 'Data' is optional depending on the configuration.

Event	defined as
	Cardnum and data formatting depends on configuration

Configuration

conf id	value format	example	explanation
/dev/mifare/device	'/dev/<dev>'	"/dev/ttyS2"	Don't touch this
/dev/mifare/key_A	<nibble>12	"FFFFFFFFFFFFFF"	
/dev/mifare/ relevant_sectors	[<n>[,<n>]*]	"1,15"	Comma separated list of the sectors to be read. Sector data is send in this order.
/dev/mifare/ prevent_duplicate_scan_timeout	n	3	A detected card will be ignored when it was the same as the last scanned mifare card within n seconds ago.
/dev/mifare/ cardnum_format	"binary" "hexadecimal"	"hexadecimal"	The cardnum is formatted according this parameter: 'binary': data is send as it is (null characters are send as null characters, etc) 'hexadecimal': all characters are transmitted in their hexadecimal value formatted as 2 nibbles for each character.

conf id	value format	example	explanation
/dev/mifare/msg/ incomplete_scan/ text	<text>	"Wait for beep \nwhen scanning"	The error message that is displayed when a card scan was incomplete.
/dev/mifare/msg/ access_violation/ text	<text>	"Card access \ndenied"	The message that is displayed when card access was denied.

8.2. Touch Screen

This document describes the requirements and design for how the NQuire application should handle the touchscreen functionality.

8.2.1. Introduction

Some customers require a possibility to interact with the NQuire. An example of this could be an application of the NQuire in a casino for upgrading a value card.

Chosen is to have a touchscreen for this. The NQuire should still remain a dumb device, knowing nothing of the application it serves. This means it will only offer the possibility to use the touchscreen. The actual application giving the touch-screen-input meaning, is to be in the customers server.

8.2.2. Hardware

This touchscreen has a capacitive matrix of 4x4 touch-area ('keys').

8.2.3. Constraints

It is not possible to use an external scanner on an NQuire with a touchscreen.

This is a driver constraint: the touchscreen outputs via STDIN as well as the external scanner. Both datastreams will be mixed, which will give inconclusive results.

Therefore, external scanner input is ignored when a touchscreen is detected.

8.2.4. Design

Escape command interface

The following extra escape commands are to be implemented:

- \xf0 <image name> \x03 : show image
 - a. position image with \x2c (set pixel position)
 - b. only gif images allowed
- \xf2 <name released> \x0d <name pressed> \x0d <pos key-id> <coupled to key-id>n \x03 : relate image to touch-key
 - a. When only 1 key is coupled, usually <pos key-id> == <coupled to key-id>
 - b. the name the name of the gif-image without the .gif extension.
 - c. it is possible to couple an image to more than 1 touch-key, simply by specifying more than 1 key-id.
 - d. positions range from [0..f], starting left top:

0	1	2	3
4	5	6	7
8	9	a	b
c	d	e	f

- \xf3 : show welcome text
 - a. This forces the welcome text to be shown without delay.
 - b. All touch-key to image bindings are released.
- \xf5 : clear text layer only (images stay on the screen)

a command format see [display touchscreen button picture](#)

Image upload

Images can be uploaded with ftp. For this, the folder 'img' is available when the ftp-user is logged in. Image upload is only available when an sd-card is inserted. The images are directly stored on the sd-card (mount --bind). As such, the size of the sd-card limits the number of images.

When no SD card is present, it is decided that it should still be possible to store a small amount of images. For this, a small separate partition is created which is mounted instead of the sd-card to /home/ftp/img. (it is not possible to just use the remaining space of the jffs2 partition because an ftp-user could then potentially crash the system by uploading files until no space is left on the jffs2 partition since that is the same partition as occupied by OS and application files).

Note that there can be serious performance degradation when there are a lot of images (thousands?). The ftp-user is responsible for managing those images.

Touch-key-press event messages

```
packet := prefix value <CR>
prefix := "K"
value  := { {'0' .. 'f'} [<filename of image>] } | 'T'
```

Note that:

- the prefix is configurable in the webui.
- the value always sends the touched key, followed by the filename of the image when an image was coupled to it. A 'T' is send on a timeout.
- When a touch-key is pressed without having an image coupled to it (and /dev/touch16/send_active_keys_only = true), then only the prefix followed by the key-value is send.

Chapter 9. Demo

This demo program is a TCP and UDP server that is used with Newland NQuire 200 to demonstrate how NQuire 200 works. You can know the communication protocol between NQuire and the server.

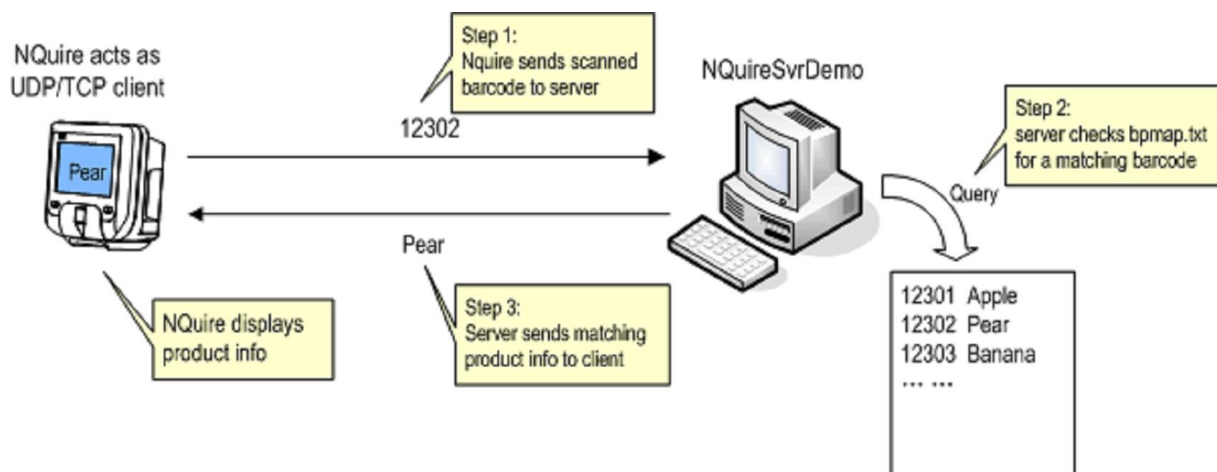
This program is developed as example only. Don't use this program for production purpose.

9.1. Applicable devices and environment

NQuireSvrDemo itself can be run on Windows 2000/XP/Vista/7/8/10(called server PC below). The server PC and the NQuire must have an IP network connection so that network data can be sent and received between the two. This demo program has been tested to work with NQuire application version 2.0 .

9.2. How to run the demo

Working procedure of NQuireSvrDemo is shown in the following figure: Preparation:



Preparation:

- Find out the IP address of your PC on which NQuireSvrDemo will run.
- Find out the IP address of NQuire.
- From your PC, use ping command to confirm NQuire is reachable on your network.
- Get some barcodes at hand to be scanned by NQuire later.

Configure NQuire side:

- On your PC, use web browser to access NQuire's configuration.
- On Network section of the web configuration, under NQuire protocol settings sub-section,

set Mode to UDP or TCP client.

and set Remote IP address to your PC's IP address.

Leave UDP port and TCP port at their defaults, 9000 and 9101 respectively. You can set the port as you will, if so, you should change the port parameters for NQuireSvrDemo accordingly.

NQuire protocol settings

UDP port	<input type="text" value="9000"/>
TCP port	<input type="text" value="9101"/>
Mode	<input type="text" value="UDP"/> ▼
Remote IP address	<input type="text" value="192.168.1.201"/>

- Click "Apply settings" to save your changes above.



If you set Mode=UDP with UDP port 9000, NQuire will send data to PC's UDP port 9000 and receive data arriving at its own UDP 9000 port.

Configure PC side:

- Prepare a bmap.txt in the same directory as NQuireSvrDemo.exe. bmap.txt provides barcode-to-product mapping for the server program. bmap.txt consists of one or more text lines, each line starts with a barcode string, then spaces/tabs, then the product info. If a barcode is received by the server program and the barcode exists in bmap.txt, the corresponding product info string will be sent back to the client(i.e to NQuire).

You can use the bundled bmap.txt.sample file as a sample of your bmap.txt. But remember to replace the sample barcode strings with the actual barcodes at your hand.

- Open a command prompt window(the console window), and cd to the directory where NQuireSvrDemo.exe resides.

- Run NQuireSvrDemo without any parameter, you can get help message about its parameter.
- Now, run NQuireSvrDemo with the following parameters:

```
NQuireSvrDemo 9000,9101 0 log
```

then NQuireSvrDemo will start to listen to NQuire's query, on UDP port 9000 and on TCP port 9101.

Console window displays something like:

```
Demo server program for Newland NQuire
Program compile date: Aug 4 2010 10:45:53
bpmmap.txt loaded, 4 items in product list.
Configuration:
TCP clients will be kept until explicitly requested.
received barcode data will be dumped to stderr.
(Press ESC to quit.)
Start listening on UDP port 9000.
Start listening on TCP port 9101.
```

Now you can start scanning a barcode with NQuire. If no error occurs, you'll see:

1. NQuireSvrDemo prints on console window the barcode 1. it receives from NQuire.
2. NQuire display on its screen the product info corresponding to that barcode. If no matching barcode found in bpmmap.txt, the server will tell NQuire to display "No such product!".

If network was not setup correctly, NQuire will not be able to receive any prompt response after scanning a barcode, in this case, NQuire displays an error message "Please Ask for Assistance".

9.2.1. More details

For a bpmmap.txt line like this,

```
12302 Pear
```

On server's receiving 12302 , it actually tells NQuire to display two lines. First line is the barcode itself(this is the hard-coded behavior in NQuireSvrDemo); the second line is the product info text following 12302 (Pear in this case).

In order to customize the display of product info, you can consult NQuire's manual to utilize NQuire special commands to fine tune the display. For example, if you don't want the echoed barcode and want Pear to be displayed at center position, you can write the text line as

```
12302 <ESC>$<ESC>.<34>Pear<03>
```

- <ESC>\$ clears the screen, thus clear the echoed barcode.
- <ESC>.<34> means aligning text at center of NQuire screen. <03> is required to close the text align command.

9.2.2. Special token in bmap.txt

If a barcode is found in bmap.txt, the bytes(so-called product info text) after [the barcode string and the separating spaces/tabs] in the very line are sent as TCP bytes to the client. But there are a few special tokens which will get replaced before sending.

- <ESC> will be replaced with one byte 0x1B .
- Text token like <XY> (where X, Y are both alphanumeric character, i.e. 0-9, A-F, a-f) will be replaced with byte whose hex value is XY . For example, <34> means a byte with value 0x34, writing '<34>' is the same as writing '4' , <0D> is carriage return. Tip: <80> can be used to denote the Euro(€) sign when NQuire is configured to use ibm852 charset.
- <00> is special, and it will be not be replaced.

Example, for a line

```
12302 <ESC>$<ESC>.<34>Pear<03>
```

Product info text will be translated into UDP/TCP bytes 1B 24 1B 2E 34 50 65 61 72 03 , ten bytes total.

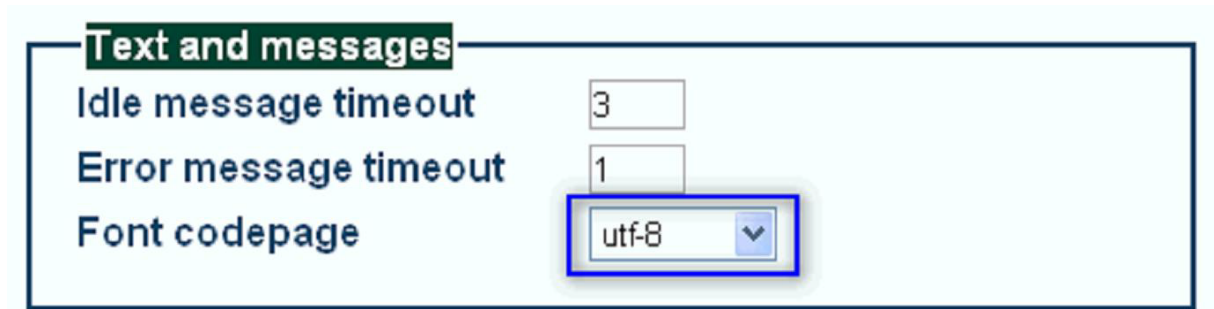
Use an asterisk as barcode string to represent "not matching barcode". That is, with a line

```
* Product not found!
```

when an input barcode cannot get a match in bmap.txt, "Product not found!" will be sent by NQuireSvrDemo .

9.2.3. How to have NQuire display Unicode characters

NQuire is capable of displaying Unicode characters, such as traditional and simplified Chinese characters. In order for it to do so, you have to configure NQuire to use utf-8 character set(charset).



More to note: SD card containing Unicode font file should be inserted into NQuire to actually display complex characters(Chinese etc). Otherwise, complex characters will be display as a small square box. It is OK to test Unicode character display with NQuireSvrDemo, but you need some special action:

1. Save bmap.1. txt in UTF-8 encoding.
2. Leave the first line of bmap.txt blank or fill some arbitrary characters on first line.
If you write a barcode at first line, that barcode will be mixed with UTF-8 BOM bytes(EF BB BF) and fail to get recognized by NQuireTcpsvrDemo.
3. Start adding your barcode at second line of bmap.txt.

9.2.4. Some hints

NQuireSvrDemo is a concurrent server, that is, it can serve multiple TCP connections and UDP clients at the same time. Press ESC key on the console window and wait for one second to quit NQuireSvrDemo gracefully. Since NQuireSvrDemo v1.3, whenever you changes bmap.txt, NQuireSvrDemo will detect that change and reload it automatically. When this occurs, you'll see a text line on the console window saying:

```
INFO: bmap.txt change detected and reloaded(4 items in list).
```

9.2.5. A demo for picture display in bmap.txt

```
#Demo display image
```



```
%A2384066542A <ESC><24><ESC><F0>ok.gif<03>
A238406654286 <ESC><24><ESC><F0>reset.gif<03>
```

9.2.6. A demo database for touch screen in bmap.txt

```
#Demo database for touch screen
B9787810454032 <ESC><42><30><ESC><24>RED WINE<ESC><2C><30><50>300
Euros<ESC><F2>1.gif<0D><0D>77a<03><ESC><F2>2.gif<0D><0D>bb<03>
K71.gif <ESC><24>50% off<ESC><2C><30><50>if you buy 2
bottles<03><ESC><F2>1.gif<0D><0D>77a<03><ESC><F2>2.gif<0D><0D>bb<03><ESC><F2>cancel.gif
Kb2.gif <ESC><24>Ask David for more
discounts<03><ESC><F2>1.gif<0D><0D>77a<03><ESC><F2>2.gif<0D><0D>bb<03><ESC><F2>cancel.
Kfcancel.gif <ESC><42><30><ESC><24>RED WINE<ESC><2C><30><50>300
Euros<ESC><F2>1.gif<0D><0D>77a<03><ESC><F2>2.gif<0D><0D>bb<03>
##KQ <ESC><40>
##KT <ESC><F3>
* <ESC><24><ESC><2E><34>(No such product!)<03>
```

9.3. Known problems

Program behaviors that are by design:

1. NQuire cannot send 4-byte sub-string "<34>" etc to client, because "<34>" is used to denote one byte of value 0x34.
2. The above limitations help keep bmap.txt simple to compose. Total number of items in bmap.txt is limited to 1000.

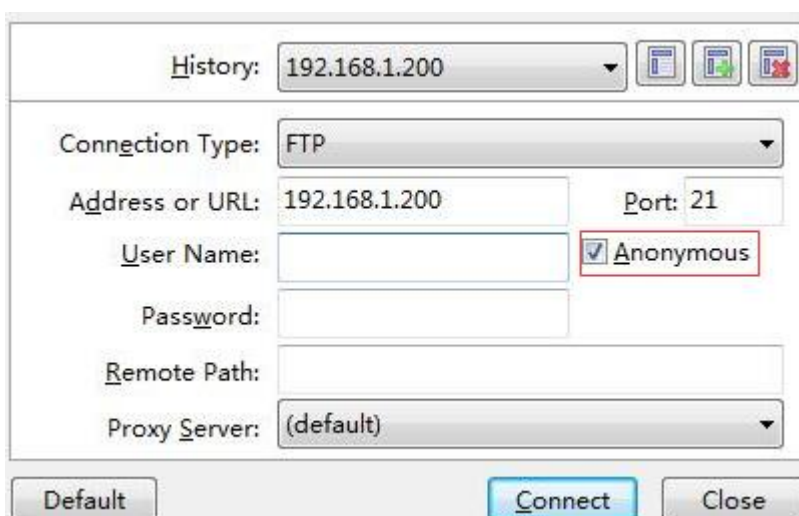
Chapter 10. Software Update

10.1. Upgrade to latest NQuire firmware using FTP

1. Startup your FTP service/program (i.e. flashfxp, filezilla, etc.);
2. Connect to the NQuire (192.168.1.200);
3. Click on the firmware stored on your pc (cit-firmware-2.0.0.build.r1104-1b8a89f86aba776969feffc67f55ab75.image do NOT rename the file) and select upload; The NQuire will show (after approx 10 seconds) uploading firmware on the screen;
4. After uploading it will shortly say uploading ok and it will restart itself (this might take a while, up to 1 min.). Do NOT unplug the power of the NQuire at any time!

After having successfully upgraded the NQuire firmware, you can now upgrade the 1D OR 2D scan engine inside the NQuire:

1. Connect to the NQuire (192.168.1.200) by FTP;
2. Click on the firmware stored on your pc (cit-em1300kernelxxx.image (231 models) OR cit-em2027kernelxxx.image and cit-em2027appxxx.image (232 models) do NOT rename the file) and select upload;
3. After uploading it will restart itself after approx. 1 min. Do NOT unplug the power of the NQuire at any time until it has indeed rebooted!



Anonymous (No User Name and Password)

Chapter 11. Summary

The NQuire customer information terminal is designed to read/scan, inform and interact with your customer. It is excellent for communicating prices, product information and loyalty points. This small and attractive information terminal reads multiple data carriers; from 1D EAN/UPC barcodes to complex 2D barcodes of mobile phone displays. It is even possible to equip the NQuire with a RFID reader. The NQuire complies with standard VESA 75 brackets enabling easy mounting on shelves and walls. Furthermore, it is possible to add USB peripherals to expand this solution with a hand held scanner for scanning large objects, a magnetic stripe reader and more. The NQuire supports various networking options: 10/100Mbps Ethernet, WiFi 802.11b/g and Power-over- Ethernet so it can be easily integrated into your existing wireless or wired LAN. The NQuire 200/230 can be used for various applications such as price checking, product information inquiries, access control, mobile barcode/coupon/ticket validations and more...

Chapter 12. Tell Us What You Think...

We'd like to know what you think about this Manual. Any suggestion? Please mail to: liuys@nlscan.com¹ (Yuansheng Liu). We will be glad to talk with you.

¹ <mailto:liuys@nlscan.com>