

# YNQ Printing Reference

**YNQ 1.5.0**

## Table of Contents

<b>1</b>	<b>REFERENCED DOCUMENTS .....</b>	<b>4</b>
<b>2</b>	<b>INTRODUCTION.....</b>	<b>5</b>
<b>3</b>	<b>ARCHITECTURE OF NQ PRINTING.....</b>	<b>6</b>
<b>4</b>	<b>PRINTING PROCESS BASICS .....</b>	<b>7</b>
4.1	ADDING PRINTER .....	8
4.2	PRINTER INSTALLATION .....	9
4.3	PRINTER MANAGEMENT .....	10
4.4	JOB MANAGEMENT .....	11
<b>5</b>	<b>REFERENCED DEFINITIONS.....</b>	<b>15</b>
5.1	CONSTANTS .....	16
5.1.1	<i>DeviceMode Flags Bits</i> .....	16
5.1.2	<i>Printer Status Flags</i> .....	16
5.1.3	<i>Printer Attribute Flags</i> .....	17
5.1.4	<i>Print Job Status Flags</i> .....	18
5.1.5	<i>Print Job Priorities</i> .....	19
5.1.6	<i>Printer Control Commands</i> .....	19
5.1.7	<i>Print Job Control Commands</i> .....	19
5.1.8	<i>Form Types</i> .....	20
5.1.9	<i>Driver OS Version</i> .....	20
5.2	STRUCTURES .....	21
5.2.1	<i>SYDeviceMode</i> .....	22
5.2.2	<i>SYPrinterDriver</i> .....	26
5.2.3	<i>SYPrinterInfo</i> .....	29
5.2.4	<i>SYPrintJobInfo</i> .....	32
5.2.5	<i>SYPrintSize</i> .....	34
5.2.6	<i>SYPrintRect</i> .....	35
5.2.7	<i>SYPrintFormInfo</i> .....	36
<b>6</b>	<b>NQ PRINTING API REFERENCE .....</b>	<b>37</b>
6.1	SYPRINTERHANDLE .....	38
6.2	SYISVALIDPRINTER .....	39
6.3	SYINVALIDATEPRINTER .....	40
6.4	SYGETPRINTERHANDLE .....	41
6.5	SYGETPRINTERINFO .....	42
6.6	SYSETPRINTERINFO .....	43
6.7	SYGETPRINTERDRIVER .....	44
6.8	SYPRINTERGETSECURITYDESCRIPTOR .....	45
6.9	SYPRINTERSETSECURITYDESCRIPTOR .....	46
6.10	SYSTARTPRINTJOB .....	47
6.11	SYENDPRINTJOB .....	48
6.12	SYSTARTPRINTPAGE .....	49
6.13	SYENDPRINTPAGE .....	50
6.14	SYWRITEPRINTDATA .....	51

6.15	SYGETPRINTJOBById .....	52
6.16	SYGETPRINTJOBIdByINDEX .....	53
6.17	SYGETPRINTJOBINDEXById .....	54
6.18	SYGETPRINTFORM .....	55
6.19	SYCONTROLPRINTER.....	56
6.20	SYCONTROLPRINTJOB .....	57
6.21	SYINITPRINTERS.....	58
6.22	SYSHUTDOWNPRINTERS .....	59
<b>7</b>	<b>IMPLEMENTATION NOTES .....</b>	<b>60</b>
7.1	USING DEVMODE STRUCTURE .....	61
7.2	USING SECURITY DESCRIPTORS.....	62
<b>8</b>	<b>SAMPLE SPOOLER .....</b>	<b>63</b>
<b>9</b>	<b>DIRECT SPOOLER .....</b>	<b>64</b>
9.1	REAL MODE.....	65
9.2	SIMULATED MODE.....	66
9.3	JOB MANAGEMENT.....	67
9.4	CONFIGURATION.....	68

## **1 Referenced Documents**

- [1] NQ Server Integration and Porting Guide
- [2] NQ User's Guide
- [3] NQ Library Reference folder
- [4] NQ SDLIB Reference

## 2 Introduction

This document describes the NQ Printing API. Printing API is an instrument for creating a print spooling system as an YNQ Server back-end.

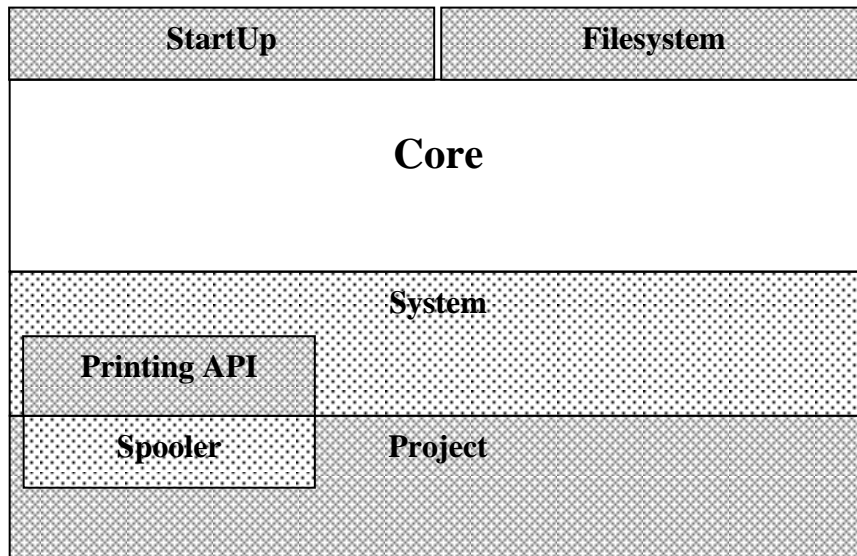
Configuring YNQ to include printing capabilities is described in [1] . This document a supplement to the [1] document.

NQ Server software defines NQ Printing API while implementing this API is a customer's role

YNQ software includes a sample implementation of a spooler. This implementation is described in section 6.5.

### 3 Architecture of NQ Printing

Printing API assumes that the target software features an actual spooling system while this API is a layer between YNQ Server and this spooling system. This approach is illustrated on Figure 1.



**Figure 2. NQ Software Vertical Decomposition**

Here “Printing API” stands for a set of data types, constants and function prototypes that designates the convention a spooler component should answer. “Spooler” is the component that implements this API. Spooler is not a part of NQ software and developing it is a customer’s responsibility.

Printing Services require NQ to be compiled with SPOOLSS protocol support. This functionality is controlled by UD\_CS\_INCLUDERPC\_SPOOLSS compile item parameter (see [1] ).

## 4 Printing Process Basics

This section designates print transactions and those SY functions that are called during those truncations. We call such calls – “indirect calls” since the respective functions are called by NQ core as a result of SMB/CIFS protocol transactions.

Each print transaction starts from indirectly calling `syGetPrinterHandle`. This call establishes correspondence between printer name and a printer handle (see 6.1).

The Spooler component is responsible for the following transaction groups:

- Adding printer to NQ server
- Printer installation
- Printer management
- Job management

#### **4.1 Adding printer**

Printer can be added by calling NQ Server API function *nqAddShareA()* (see [3] ).

Sample code looks like:

```
nqAddShareA("printer", "/dev/usb/lp0", TRUE, "Shared printer", "");
```

where the arguments go as follows:

1. A printer share name as it will be visible by SMB/CIFS clients.
2. Actual local device path.
3. Must be TRUE for printer shares.
4. Text of share comment.
5. Reserved (currently unused).

This function returns 0 if printer installs successfully.



## **4.2 Printer Installation**

This process occurs on the client side once for each client willing to use a printer connected to NQ Server. The printer installation process may occur in two shapes:

1. NQ Server reports the printer type and the location of the relevant driver. The client computer downloads driver from NQ Server.
2. NQ Server reports the printer type and characteristics so that a client may uniquely identify it. Then user installs printer driver locally on the client from a media.

Both methods indirectly call *syGetPrinterInfo()* (see 6.6) and *syGetPrinterDriver()* (see 6.7) functions.

### **4.3 Printer Management**

When a client computer displays printer information the following functions are indirectly called:

```
syGetPrinterInfo  
syGetPrintForm  
syPrinterGetSecurityDescriptor
```

When a client computer modifies printer information the following functions are indirectly called:

```
sySetPrinterInfo  
syPrinterSetSecurityDescriptor
```

When client computer changes printer statues it indirectly calls the following functions:

```
syControlPrinter
```

#### **4.4 Job Management**

When client computer displays printer information the following functions are indirectly called:

- `syGetPrintJobById`
- `syGetPrintJobIdByIndex`
- `syGetPrintJobIndexById`

Last two functions establish correspondence between job index and job ID. Job index is a zero-based number designating job position in the print queue. When spooler reorganizes its queue job index of a particular job may change so that all jobs in the queue have subsequent number of their position in the queue.

Job ID is a number assigned to a job when it is created and it remains the same through over job lifecycle. No two existing jobs may have the same ID. However, ID can be reused.

When client computer submits a print job the following functions are indirectly called:

- `syStartPrintJob` – one call per a job
- `syStartPrintPage` – one call one call for each page. When document is not divided into pages, this call may be omitted
- `syWritePrintData` – one or more calls for each page
- `syEndPrintPage` – this call matches corresponding `syStartPrintPage`
- `syEndPrintJob` – this call matches corresponding `syStartPrintJob`

When client computer changes job statues it indirectly calls the following functions:

## 4.5 *syControlPrintJob*

### Prototype

```
NQ_STATUS syControlPrintJob(
    SYPrinterHandle handle,
    NQ_UINT32 jobId,
    NQ_UINT32 command
);
```

### Description

Perform a job command

### Parameters

handle	IN
Printer handle	
jobId	IN
Job ID	
command	IN
Job command code. See 5.1.7 for possible codes.	

### Return Value

SUCCESS or FAIL

### Notes

Job ID is the number returned by *syStartPrintJob()* (see 6.10).

## 4.6 *syInitPrinters*

### Prototype

```
NQ_BOOL syInitPrinters (  
    SYPrinterPrepareSendLateResponse  
    printerPrepareSendLateWriteResponse,  
    NQ_COUNT responseContextSize  
);
```

### Description

Initialize the printer module

### Parameters

<code>printerPrepareSendLateWriteResponse</code>	IN
	Pointer to callback for sending delayed response
<code>responseContextSize</code>	IN
	Delayed response size to be allocated when needed

### Return Value

SUCCESS or FAIL

### Notes

NQ Core responsibility to initialize the printer module with the callback of type:  
typedef void (\*SYPrinterPrepareSendLateResponse)(SYPrinterJob \*,  
NQ\_INT);

## **4.7 *syShutdownPrinters***

Prototype

```
void syShutdownPrinters(void);
```

Description

Shutdown the printer module

Parameters

None

Return Value

None

Notes

This call deallocated the delayed response buffers per job

## **5 Referenced Definitions**

This section designates constants and data types that are referenced in the NQ Printing API. These definitions are a part of NQ Core software and they should not be modified during NQ Printing API implementation.

## 5.1 Constants

This section designates those constants that are referenced in Printing API. All relevant constants are defined in the file *SYCOMMON.H*.

### 5.1.1 DeviceMode Flags Bits

The following constants define the which fields of the Device Mode structure (see 5.2.1) are set. This structure allows partial information been provided. Constants below when set as bits of the Flags bit-mask field of this structure designate those values that are currently set. Name of a constant corresponds to the field name in the structure.

Name	Value
SY_DEVICEMODE_ORIENTATION	0x00000001
SY_DEVICEMODE_PAPERSIZE	0x00000002
SY_DEVICEMODE_PAPERLENGTH	0x00000004
SY_DEVICEMODE_PAPERWIDTH	0x00000008
SY_DEVICEMODE_SCALE	0x00000010
SY_DEVICEMODE_POSITION	0x00000020
SY_DEVICEMODE_NUP	0x00000040
SY_DEVICEMODE_COPIES	0x00000080
SY_DEVICEMODE_DEFAULTSOURCE	0x00000100
SY_DEVICEMODE_PRINTQUALITY	0x00000400
SY_DEVICEMODE_COLOR	0x00000800
SY_DEVICEMODE_DUPLEX	0x00001000
SY_DEVICEMODE_YRESOLUTION	0x00002000
SY_DEVICEMODE_TTOPTION	0x00004000
SY_DEVICEMODE_COLLATE	0x00008000
SY_DEVICEMODE_FORMNAME	0x00010000
SY_DEVICEMODE_LOGPIXELS	0x00020000
SY_DEVICEMODE_BITSPERPEL	0x00040000
SY_DEVICEMODE_PELWIDTH	0x00080000
SY_DEVICEMODE_PELHEIGHT	0x00100000
SY_DEVICEMODE_DISPLAYFLAGS	0x00200000
SY_DEVICEMODE_DISPLAYFREQUENCY	0x00400000
SY_DEVICEMODE_ICMMETHOD	0x00800000
SY_DEVICEMODE_ICMINTENT	0x01000000
SY_DEVICEMODE_MEDIATYPE	0x02000000
SY_DEVICEMODE_DITHERTYPE	0x04000000
SY_DEVICEMODE_PANNINGWIDTH	0x08000000
SY_DEVICEMODE_PANNINGHEIGHT	0x10000000

### 5.1.2 Printer Status Flags

The following constants designate printer status flags. Printer status may report any reasonable combination of these flags.

Name	Value
SY_PRINTERSTATUS_PAUSED	0x00000001
SY_PRINTERSTATUS_ERROR	0x00000002



SY_PRINTERSTATUS_PENDINGDELETION	0x00000004
SY_PRINTERSTATUS_PAPERJAM	0x00000008
SY_PRINTERSTATUS_PAPEROUT	0x00000010
SY_PRINTERSTATUS_MANUALFEED	0x00000020
SY_PRINTERSTATUS_PAPERPROBLEM	0x00000040
SY_PRINTERSTATUS_OFFLINE	0x00000080
SY_PRINTERSTATUS_IOACTIVE	0x00000100
SY_PRINTERSTATUS_BUSY	0x00000200
SY_PRINTERSTATUS_PRINTING	0x00000400
SY_PRINTERSTATUS_OUTPUTBINFULL	0x00000800
SY_PRINTERSTATUS_NOTAVAILABLE	0x00001000
SY_PRINTERSTATUS_WAITING	0x00002000
SY_PRINTERSTATUS_PROCESSING	0x00004000
SY_PRINTERSTATUS_INITIALIZING	0x00008000
SY_PRINTERSTATUS_WARMINGUP	0x00010000
SY_PRINTERSTATUS_TONERLOW	0x00020000
SY_PRINTERSTATUS_NOTONER	0x00040000
SY_PRINTERSTATUS_PAGEPUNT	0x00080000
SY_PRINTERSTATUS_USERINTERVENTION	0x00100000
SY_PRINTERSTATUS_OUTOFMEMORY	0x00200000
SY_PRINTERSTATUS_DOOROPEN	0x00400000
SY_PRINTERSTATUS_SERVERUNKNOWN	0x00800000
SY_PRINTERSTATUS_POWERSAVE	0x01000000

### 5.1.3 Printer Attribute Flags

The following constants designate printer attributes (capabilities). Printer information structure may report any reasonable combination of these attribute flags. Remote management might force changing some statuses. The spooling/printing system is not required to necessary accept this attempt.

Name	Value	Comment
SY_PRINTERATTR_QUEUED	0x0001	Spools first, then prints
SY_PRINTERATTR_DIRECT	0x0002	Jobs are printed directly, without spooling
SY_PRINTERATTR_DEFAULT	0x0004	9x/ME only - default printer
SY_PRINTERATTR_SHARED	0x0008	Shared printer
SY_PRINTERATTR_NETWORK	0x0010	Network printer
SY_PRINTERATTR_RESERVED	0x0020	
SY_PRINTERATTR_LOCAL	0x0040	Local printer
SY_PRINTERATTR_DEVQUERY	0x0080	Not supported, should be clear
SY_PRINTERATTR_KEEPCJOBS	0x0100	Do not delete jobs after printing
SY_PRINTERATTR_LIFO	0x0200	jobs are printed in back order
SY_PRINTERATTR_OFFLINE	0x0400	Not supported, should be clear
SY_PRINTERATTR_BIDI	0x0800	Not supported, should be clear
SY_PRINTERATTR_RAWONLY	0x1000	Only raw data may be spooled
SY_PRINTERATTR_PUBLISHED	0x2000	Printer is published in the directory

SY_PRINTERATTR_EXPAND	0x4000	A print provider can set this flag as a hint to a calling application to enumerate this object further if default expansion is enabled. For example, when domains are enumerated, a print provider might indicate the user's domain by setting this flag.
SY_PRINTERATTR_CONTAINER	0x8000	If this flag is set, the printer object may contain enumerable objects. For example, the object may be a print server that contains printers.
SY_PRINTERATTR_ICON1	0x10000	Indicates that, where appropriate, an application should display an icon identifying the object as a top-level network name, such as Microsoft Windows Network.
SY_PRINTERATTR_ICON2	0x20000	Indicates that, where appropriate, an application should display an icon that identifies the object as a network domain.
SY_PRINTERATTR_ICON3	0x40000	Indicates that, where appropriate, an application should display an icon that identifies the object as a print server.
SY_PRINTERATTR_ICON4	0x80000	Reserved.
SY_PRINTERATTR_ICON5	0x100000	Reserved.
SY_PRINTERATTR_ICON6	0x200000	Reserved.
SY_PRINTERATTR_ICON7	0x400000	Reserved.
SY_PRINTERATTR_ICON8	0x800000	Indicates that, where appropriate, an application should display an icon that identifies the object as a printer

#### 5.1.4 Print Job Status Flags

The following constants designate status flags for a print job. Job information structure may report any reasonable combination of these flags in the status field.

Name	Value	Comment
SY_PRINTJOBSTATUS_PAUSED	0x00000001	Job is paused

SY_PRINTJOBSTATUS_ERROR	0x00000002	An error is associated with the job
SY_PRINTJOBSTATUS_DELETING	0x00000004	Job is being deleted
SY_PRINTJOBSTATUS_SPOOLING	0x00000008	Job is queued
SY_PRINTJOBSTATUS_PRINTING	0x00000010	Job is being printed
SY_PRINTJOBSTATUS_OFFLINE	0x00000020	Printer is offline
SY_PRINTJOBSTATUS_PAPERPROBLEM	0x00000040	Paper jam or out of paper
SY_PRINTJOBSTATUS_PRINTED	0x00000080	Job has been printed
SY_PRINTJOBSTATUS_DELETED	0x00000100	Job has been deleted
SY_PRINTJOBSTATUS_BLOCKED	0x00000200	The driver cannot print the job
SY_PRINTJOBSTATUS_USERINTERVENTION	0x00000400	User intervention required
SY_PRINTJOBSTATUS_RESTART	0x00000800	Job has been restarted

#### 5.1.5 Print Job Priorities

The following constants designate print job priority limits and defaults. Job information structure reports one of such values in the priority field.

Name	Value	Comment
SY_PRINTJOB_MINPRIORITY	1	Minimum priority
SY_PRINTJOB_MAXPRIORITY	99	Maximum priority
SY_PRINTJOB_DEFPPRIORITY	1	Default priority

#### 5.1.6 Printer Control Commands

The following constants designate command codes for a printer.

Name	Value	Comment
SY_PRINTERCONTROL_PAUSE	1	Pause printer
SY_PRINTERCONTROL_RESUME	2	Resume printer
SY_PRINTERCONTROL_PURGE	3	Purge all jobs
SY_PRINTERCONTROL_SETSTATUS	4	Force printer status

#### 5.1.7 Print Job Control Commands

The following constants designate command codes for a print job.

Name	Value	Comment
SY_PRINTJOB_COM_PAUSE	1	Pause a job
SY_PRINTJOB_COM_RESUME	2	Resume a paused job

SY_PRINTJOB_COM_CANCEL	3	Cancel a job
SY_PRINTJOB_COM_RESTART	4	Restart the print job. A job can only be restarted if it was printing
SY_PRINTJOB_COM_DELETE	5	Delete the print job
SY_PRINTJOB_COM_SENDDTOPRINTER	6	Used by port monitors to end the print job
SY_PRINTJOB_COM_LASTPAGEEJECTED	7	Used by language monitors to end the print job

### 5.1.8 Form Types

The following constants designate form types.

Name	Value	Comment
SY_PRINTFORMFLAG_USER	0	User-defined
SY_PRINTFORMFLAG_BUILTIN	1	Spooler level
SY_PRINTFORMFLAG_PRINTER	2	Printer level

### 5.1.9 Driver OS Version

The following constants designate operating system the printer driver was developed for. More values may be added to this table on demand.

Name	Value	Comment
SY_PRINTEROSVERSION_WIN	3	Windows operating systems

## **5.2 Structures**

This section provides the definitions for structures used in SMB/CIFS Printing API.

### 5.2.1 SYDeviceMode

#### Prototype

```
typedef struct
{
    NQ_UINT16 specVersion;
    NQ_UINT16 driverVersion;
    NQ_UINT16 size;
    NQ_UINT16 driverExtraLength;
    const NQ_BYTE* driverExtraData;
    NQ_UINT32 fields;
    NQ_UINT16 orientation;
    NQ_UINT16 paperSize;
    NQ_UINT16 paperLength;
    NQ_UINT16 paperWidth;
    NQ_UINT16 scale;
    NQ_UINT16 copies;
    NQ_UINT16 defaultSource;
    NQ_UINT16 printQuality;
    NQ_UINT16 color;
    NQ_UINT16 duplex;
    NQ_UINT16 yResolution;
    NQ_UINT16 ttOption;
    NQ_UINT16 collate;
    const NQ_TCHAR* formName;
    NQ_UINT16 logPixels;
    NQ_UINT32 bitsPerPel;
    NQ_UINT32 pelsWidth;
    NQ_UINT32 pelsHeight;
    NQ_UINT32 displayFlags;
    NQ_UINT32 displayFrequency;
    NQ_UINT32 icmMethod;
    NQ_UINT32 icmIntent;
    NQ_UINT32 mediaType;
    NQ_UINT32 ditherType;
    NQ_UINT32 reserved1;
    NQ_UINT32 reserved2;
    NQ_UINT32 panningWidth;
    NQ_UINT32 panningHeight;
} SYDeviceMode;
```

#### Description

This structure describes printer hardware. It consists of required members and optional members. Required members are those that reside before the *fields* member. There is a bit in the *fields* member for each of the following members. A set bit means that the corresponding member carries an actual value. Some windows version uses this structure values to calculate number of pages to be printed and the raw size of the job. When spooler does not supply printer geometry values in this structure (as specified by the *fields* bits) those calculations may result in garbage values being displayed in the job list.

## Members

`specVersion`

The version of this structure. Should read 0x401

`driverVersion`

Should be the same number as in the *SYPrinterDriver* structure

`size`

For compatibility with the protocol's structure. This value is not used

`driverExtraLength`

Number of bytes in the driver extra data. Extra data is driver-specific information that might be passed to the printer job. This value may be zero

`driverExtraData`

Extra data is driver-specific information that might be passed to the printer job. This member is a pointer to a memory block of the *driverExtraLength* size. This value may be NULL.

`fields`

This member specifies which of the subsequent members contain actual data. There is a bit in the *fields* member for each of them. A set bit means that the corresponding member carries an actual value. See 5.1.1 for possible values.

`orientation`

Paper orientation in the printer as: 1 for portrait, 2 for landscape.

`paperSize`

Paper size code. A4 is designated by 9. Other codes are to be designated in the future versions of NQ Printing API

`paperLength`

Paper length in centimeters multiplied by *scale* (see below). With scale value of 100 this value should be 2970 (29.7 centimeter) for the A4 paper size.

`paperWidth`

Paper width in centimeters multiplied by *scale* (see below). With scale value of 100 this value should be 2100 (21.0 centimeter) for the A4 paper size.

`scale`

Accuracy of paper dimensions in fractions of centimeter. For instance, the value of (`paperWidth * scale`) gives paper width in centimeters.

`copies`

Number of copies printed on the hardware level.  
Normally this value is set to 1.

defaultSource

This value is for future usage. In the current version of  
NQ Server software this value should be set to 15.

printQuality

Printer resolution in DPI.

color

This value is for future usage. In the current version of  
NQ Server software this value should be set to 1.

duplex

This value is for future usage. In the current version of  
NQ Server software this value should be set to 2.

yResolution

Vertical printing resolution in DPI.

ttOption

This value is for future usage. In the current version of  
NQ Server software this value should be set to 3.

collate

This value is for future usage. In the current version of  
NQ Server software this value should be set to 1.

formName

Pointer to a printable form name, e.g., - "A4"

logPixels

Number of logical pixels. This value is not used by NQ  
Server software and is passed as is to or from the printing  
system.

bitsPerPel

Number of bits per pixel. This value is not used by NQ  
Server software and is passed as is to or from the printing  
system.

pelsWidth

Pixel width. This value is not used by NQ Server software  
and is passed as is to or from the printing system.

pelsHeight

Pixel height. This value is not used by NQ Server software  
and is passed as is to or from the printing system.

displayFlags

This value is for future usage. In the current version of  
NQ Server software this value should be set to 1.

displayFrequency

This value is for future usage. In the current version of  
NQ Server software this member can contain any value.



icmMethod

This value is for future usage. In the current version of NQ Server software this value should be set to 1.

icmIntent

This value is for future usage. In the current version of NQ Server software this value should be set to 2.

mediaType

This value is for future usage. In the current version of NQ Server software this value should be set to 0x10C.

ditherType

This value is for future usage. In the current version of NQ Server software this value should be set to 0xFFFFFFFF.

reserved1

This value is for future usage. In the current version of NQ Server software this member can contain any value.

reserved2

This value is for future usage. In the current version of NQ Server software this member can contain any value.

panningWidth

This value is for future usage. In the current version of NQ Server software this member can contain any value.

panningHeight

This value is for future usage. In the current version of NQ Server software this member can contain any value.

## 5.2.2 SYPrinterDriver

### Prototype

```
typedef struct
{
    NQ_UINT32 osVersion;
    const NQ_TCHAR* name;
    const NQ_TCHAR* driverPath;
    const NQ_TCHAR* dataFile;
    const NQ_TCHAR* configFile;
    const NQ_TCHAR* helpFile;
    const NQ_TCHAR** dependentFiles;
    const NQ_TCHAR* monitorName;
    const NQ_TCHAR* defaultDataType;
    const NQ_TCHAR** previousNames;
    NQ_UINT32 driverDate;
    NQ_UINT32 driverVersions[2];
    const NQ_TCHAR* manufacturer;
    const NQ_TCHAR* manufacturerURL;
    const NQ_TCHAR* hardwareID;
    const NQ_TCHAR* provider;
    NQ_UINT32 attributes;
    NQ_UINT32 configVersion;
    NQ_UINT32 driverVersion;
} SYPrinterDriver;
```

### Description

This structure describes printer driver. It contains necessary information for installing the driver. All files referenced in this structure should reside in the printer share.

### Members

osVersion

Version of the operating system this driver is developed for see 5.1.9 for possible values.

name

Pointer to the printer name. This should be a null-terminated string. Driver is identified by its name.

driverPath

Pointer to the null-terminated string designating a full path to the driver file. This path should start from the share name prefixed by two backslashes.

dataFile

Pointer to the null-terminated string designating a full path to the data file. This path should start from the share name prefixed by two backslashes.

configFile

Pointer to the null-terminated string designating a full path to the driver configuration utility file. This path

should start from the share name prefixed by two backslashes.

helpFile

Pointer to the null-terminated string designating a full path to the driver help file. This path should start from the share name prefixed by two backslashes.

dependentFiles

Double pointer to the list of additional files required by the driver. This list should be an array of pointers to null-terminated strings each one designating a full path to an additional file. A path should start from the share name prefixed by two backslashes. The list of pointers should be terminated with a NULL pointer. This list can be empty with only the terminating NULL pointer.

monitorName

Pointer to the null-terminated string with a name of the language monitor.

defaultDataType

Pointer to the null-terminated string with a name of the default data type. This member can contain a NULL pointer.

previousNames

Double pointer to the list of previous names for the driver. This list should be an array of pointers to null-terminated strings each one designating one previous name. The list of pointers should be terminated with a NULL pointer. This list can be empty with only the terminating NULL pointer.

driverDate

Driver release date in Unix format. This member can contain zero.

driverVersions

This array contains major and minor version of the driver

manufacturer

Pointer to the null-terminated string with a name of the driver manufacturer. This member can contain a NULL pointer.

manufacturerURL

Pointer to the null-terminated string with the URL of the driver manufacturer's WEB site. This member can contain a NULL pointer.

hardwareID

Pointer to the null-terminated string with a manufacturer's identification of the driver in a printable form.

provider

Pointer to the null-terminated string with a name of the driver provider.

attributes

This value is for future usage. In the current version of NQ Server software this value should be set to zero.

configVersion

This value is for future usage. In the current version of NQ Server software this value should be set to zero.

driverVersion

This value is for future usage. In the current version of NQ Server software this value should be set to zero.

### 5.2.3 SYPrinterInfo

#### Prototype

```
typedef struct
{
    NQ_UINT32 flags;
    NQ_TCHAR* portName;
    NQ_TCHAR* driverName;
    NQ_TCHAR* comment;
    NQ_TCHAR* location;
    SYDeviceMode devMode;
    NQ_TCHAR* sepFile;
    NQ_TCHAR* printProcessor;
    NQ_TCHAR* dataType;
    NQ_TCHAR* parameters;
    NQ_UINT32 attributes;
    NQ_UINT32 priority;
    NQ_UINT32 defaultPriority;
    NQ_UINT32 startTime;
    NQ_UINT32 untilTime;
    NQ_UINT32 status;
    NQ_UINT32 cJobs;
    NQ_UINT32 totalJobs;
    NQ_UINT32 totalBytes;
    NQ_UINT32 globalCounter;
    NQ_UINT32 totalPages;
    NQ_UINT16 majorVersion;
    NQ_UINT16 buildVersion;
    NQ_UINT32 sessionCounter;
    NQ_UINT32 printerErrors;
    NQ_UINT32 cSetPrinter;
    NQ_UINT32 averagePpm;
    NQ_UINT32 deviceNotSelectedTimeout;
    NQ_UINT32 transmissionRetryTimeout;
    NQ_BYTE* securityDescriptor;
    NQ_UINT32 securityDescriptorLength;
}
SYPrinterInfo;
```

#### Description

This structure carries printer information.

#### Members

flags

Printer flags (see 5.2.1).

portName

Pointer to the null-terminated string with the ID of the port this printer is connected over.

driverName

	Pointer to the null-terminated string with the ID of the driver this printer is controlled by. If the server does not support driver downloading this field should contain NULL.
<code>comment</code>	
	Pointer to the null-terminated string with the text of the printer description.
<code>location</code>	
	Pointer to the null-terminated string with the description of the printer location.
<code>devMode</code>	
	Pointer to the SYDeviceMode structure describing this printer hardware. See 5.2.1 for details.
<code>sepFile</code>	
	Pointer to the null-terminated string with the path to an optional separator file.
<code>printProcessor</code>	
	Pointer to the null-terminated string with an ID of expected (required) print processor. This field should read "winprint" for Windows.
<code>dataType</code>	
	Pointer to the null-terminated string with a name of the expected data type. "RAW" is the default value.
<code>parameters</code>	
	Pointer to the null-terminated string with printer-specific parameters.
<code>attributes</code>	
	Printer attributes. See 5.1.3 for possible values.
<code>priority</code>	
	Priority of the job being printed.
<code>defaultPriority</code>	
	Default job priority.
<code>startTime</code>	
	Time this printer will be available in Unix format. This value may be zero.
<code>untilTime</code>	
	Time this printer will become unavailable in Unix format. This value may be zero.
<code>status</code>	
	Printer status. See 5.1.2 for possible values.
<code>cJobs</code>	

Number of pending jobs including those that has been spooled down and are not printed yet.

totalJobs

Accumulated number of printed jobs.

totalBytes

Accumulated number of printed bytes.

globalCounter

This field is not in use in the current version of NQ Server software.

totalPages

Accumulated number of printed pages.

majorVersion

Major printer version.

buildVersion

Version of the printer firmware.

sessionCounter

This field is not in use in the current version of NQ Server software.

printerErrors

Accumulated number of print errors.

cSetPrinter

Last printer command.

averagePpm

Average number of pages printed per minute.

deviceNotSelectedTimeout

This field is not in use in the current version of NQ Server software.

transmissionRetryTimeout

This field is not in use in the current version of NQ Server software.

securityDescriptor

Pointer to the security descriptor associated with the printer.

securityDescriptorLength

Length of the security descriptor associated with the printer.

## 5.2.4 SYPrintJobInfo

### Prototype

```
typedef struct
{
    NQ_UINT32 id;
    const NQ_TCHAR* documentName;
    const NQ_TCHAR* pStatus;
    NQ_UINT32 status;
    NQ_UINT32 priority;
    NQ_UINT32 position;
    NQ_UINT32 totalPages;
    NQ_UINT32 pagesPrinted;
    NQ_UINT32 submitTime;
    NQ_UINT32 startTime;
    NQ_UINT32 untilTime;
    NQ_UINT32 time;
    NQ_UINT32 size;
    NQ_BYTE* securityDescriptor;
    NQ_UINT32 securityDescriptorLength;
    Const void * user;
}
SYPrintJobInfo;
```

### Description

This structure carries information about a print job.

### Members

id	A number uniquely identifying this job.
documentName	Pointer to the null-terminated string with the name of the document being printed.
pStatus	Pointer to the null-terminated string with the status of this job in a printable form. This value can be NULL. This member should be used only when job status cannot be covered by possible values of the <i>status</i> member.
Status	Current status of the job. Possible values are described in 5.1.4. This member is the preferred alternative to <i>pStatus</i> .
Priority	Job priority
Position	Job position in the queue
totalPages	Number of pages in the entire job.



pagesPrinted

Number of pages printed so far

submitTime

Time this job was queued

startTime

The earliest time this job can be printed

untilTime

The latest time this job can be printed

Time

Elapsed time since this job begun being printed

Size

Job size in bytes

securityDescriptor

Pointer to the security descriptor associated with the job.

securityDescriptorLength

Length of the security descriptor associated with the job.

User

User context of the job. Spooler accepts this value on *syStartPrintJob()* call (see 6.10) reports into this structure. Spooler is not expected to interpret this value.

### 5.2.5 SYPrintSize

#### Prototype

```
typedef struct
{
    NQ_UINT32 width;
    NQ_UINT32 height;
}
SYPrintSize;
```

#### Description

This structure describes printing geometry for difference occasions.

#### Members

width	
	width in thousandths of millimeters
height	
	height in thousandths of millimeters

## 5.2.6 SYPrintRect

### Prototype

```
typedef struct
{
    NQ_UINT32 left;
    NQ_UINT32 top;
    NQ_UINT32 right;
    NQ_UINT32 bottom;
}
SYPrintRect;
```

### Description

This structure describes printing geometry for difference occasions.

### Members

left	horizontal shift to the left edge in thousandths of millimeters
top	vertical shift to the top edge in thousandths of millimeters
right	horizontal shift to the right edge in thousandths of millimeters
bottom	vertical shift to the bottom edge in thousandths of millimeters

### 5.2.7 SYPrintFormInfo

#### Prototype

```
typedef struct
{
    NQ_UINT32 id;
    const NQ_TCHAR* name;
    NQ_UINT32 flags;
    SYPrintSize size;
    SYPrintRect imageableArea;
}
SYPrintFormInfo;
```

#### Description

This structure describes a print form.

#### Members

<code>id</code>	Unique number for this form
<code>name</code>	Pointer to a null-terminated string with form name
<code>flags</code>	Form type as specified in 5.1.8
<code>size</code>	Form dimensions
<code>imageableArea</code>	Print area in the form

## **6 NQ Printing API Reference**

This section designates API functions. NQ Server software contains prototypes for those functions while implementing them is a customer's role.

All API functions are prototyped in the file SYPRINTR.H.

## 6.1 ***SYPrinterHandle***

### Prototype

`SYPrinterHandle`

### Description

Data type for a printer handle

### Parameters

None

### Notes

This data type should provide unique printer identification. It can be an index in the printer table or a pointer to a structure. See also 6.2 and 6.3 for special values of this handle.

## 6.2 *syIsValidPrinter*

### Prototype

```
NQ_BOOL syIsValidPrinter(  
    SYPrinterHandle handle  
);
```

### Description

This call validates printer handle

### Parameters

handle	IN
--------	----

Printer handle to validate

### Return Value

TRUE when the handle is valid, TRUE otherwise

### Notes

This call should return TRUE for any value but the one returned by the *syInvalidatePrinter()* function (see 6.3).

### 6.3 *syInvalidatePrinter*

Prototype

```
void syInvalidatePrinter (  
    SYPrinterHandle* handle  
);
```

Description

Make printer handle invalid

Parameters

handle	OUT
Pointer to the handle	

Return Value

None

Notes

This call should set printer handle, designated by the parameter pointer, to a special value different from any valid printer handle. This value, when used as a parameter to the *syIsValidprinter()* (see 6.2) should cause a FALSE result.



## 6.4 *syGetPrinterHandle*

### Prototype

```
SYPrinterHandle syGetPrinterHandle(  
    const NQ_TCHAR* name  
);
```

### Description

Get printer handle by name

### Parameters

Name	IN
Printer name	

### Return Value

Printer handle or invalid handle

### Notes

This function should associate a unique printer handle with its name. It should find appropriate printer in a table or a list of supported printers.

## 6.5 *syGetPrinterInfo*

### Prototype

```
NQ_STATUS syGetPrinterInfo(  
    SYPrinterHandle handle,  
    SYPrinterInfo* info  
);
```

### Description

Get printer information

### Parameters

handle	IN
Printer handle	
info	IN
Buffer for information structure (see 5.2.3)	

### Return Value

SUCCESS or FAIL

### Notes

Printer information structure (see 5.2.3) contains several pointers. Those pointers when set should reference a persistent data, meaning that it should remain unchanged until the next call to the same function for the same printer. See also comment about the *DevMode* structure in 5.2.1.

## 6.6 *sySetPrinterInfo*

### Prototype

```
NQ_STATUS sySetPrinterInfo(  
    SYPrinterHandle handle,  
    const SYPrinterInfo* info  
);
```

### Description

Update printer information

### Parameters

Handle	IN
Printer handle	
Info	IN
Pointer to the information structure (see 5.2.3)	

### Return Value

SUCCESS or FAIL

### Notes

Printer information structure (see 5.2.3) contains several pointers. Those pointers when set do not reference a persistent data. This means that the implementation of this call should copy referenced data. See also comment about the *DevMode* structure in 5.2.1.

## 6.7 *syGetPrinterDriver*

### Prototype

```
NQ_STATUS syGetPrinterDriver(  
    SYPrinterHandle handle,  
    const NQ_TCHAR* os,  
    SYPrinterDriver* info  
);
```

### Description

Get information about the printer driver

### Parameters

handle	IN
Printer handle	
os	IN
Name of the operating system requesting the driver	
info	IN
Buffer for the information structure (see 5.2.2)	

### Return Value

SUCCESS or FAIL

### Notes

Driver information structure (see 5.2.2) contains several pointers. Those pointers when set should reference a persistent data, meaning that it should remain unchanged until the next call to the same function for the same printer. If the server does not support driver downloading, this call should return FAIL.

## 6.8 *syPrinterGetSecurityDescriptor*

### Prototype

```
NQ_COUNT syPrinterGetSecurityDescriptor(  
    SYPrinterHandle handle,  
    NQ_BYTE* buffer,  
    NQ_COUNT bufferLength  
);
```

### Description

Update printer security descriptor

### Parameters

handle	IN
Printer handle	
buffer	OUT
Buffer for security descriptor	
bufferLength	IN
Buffer size	

### Return Value

Actual length of the copied security descriptor FAIL

### Notes

This function should copy a security descriptor of the printer designated by the first parameter to the buffer.

## 6.9 *syPrinterSetSecurityDescriptor*

### Prototype

```
NQ_STATUS syPrinterSetSecurityDescriptor(  
    SYPrinterHandle handle,  
    const NQ_BYTE* descriptor,  
    NQ_INT length  
);
```

### Description

Update printer security descriptor

### Parameters

handle	IN
Printer handle	
descriptor	IN
Pointer to new security descriptor	
length	IN
Length of the new descriptor	

### Return Value

SUCCESS or FAIL

### Notes

This function gets either the default security descriptor or a security descriptor explicitly sent by the client machine.

## 6.10 *syStartPrintJob*

### Prototype

```
NQ_UINT32 syStartPrintJob(
    SYPrinterHandle handle,
    const NQ_TCHAR* name,
    const NQ_TCHAR* file,
    const NQ_TCHAR* type,
    const NQ_BYTE* sd,
    NQ_COUNT sdLen,
    Const void *pUser
);
```

### Description

Start new print job

### Parameters

handle	IN
Printer handle	
name	IN
Document (also job) name	
file	IN
File name	
type	IN
Data type or NULL	
sd	IN
Security descriptor	
sdLen	IN
Security descriptor length	
pUser	IN
Pointer to user context	

### Return Value

ID of the new job or zero on error

### Notes

Document name and file name are supplied for information only and it should be saved and returned on subsequent information calls (see 6.15 and 6.16).

Spooler should report user context (*pUser*) in the *user* field of the job information structure (see 5.2.4) as required by function *syGetPrintJobById()* (see 6.15). Spooler is not expected to interpret this value.

### 6.11 *syEndPrintJob*

**Prototype**

```
NQ_STATUS syEndPrintJob(  
    SYPrinterHandle handle,  
    NQ_UINT32 jobId,  
);
```

**Description**

End print job

**Parameters**

Handle	IN
Printer handle	
jobId	IN
Job ID as returned by <i>syStartPrintJob</i> (see 6.10)	

**Return Value**

SUCCESS or FAIL

**Notes**

This call signals of a normal job completion. Spooler should release job resources and remove this job from the list of jobs.



## 6.12 *syStartPrintPage*

### Prototype

```
NQ_STATUS syStartPrintPage(  
    SYPrinterHandle handle,  
    NQ_UINT32 jobId,  
);
```

### Description

Start new page for the current print job

### Parameters

handle	IN
Printer handle	
jobId	IN
Job ID as returned by <i>syStartPrintJob</i> (see 6.10)	

### Return Value

SUCCESS or FAIL

### Notes

On getting this call spooler should increment job counter for this job.

### 6.13 *syEndPrintPage*

Prototype

```
NQ_STATUS syEndPrintPage(  
    SYPrinterHandle handle,  
    NQ_UINT32 jobId,  
);
```

Description

Close current page for the current print job

Parameters

handle	IN
Printer handle	
jobId	IN
Job ID as returned by <i>syStartPrintJob</i> (see 6.10)	

Return Value

SUCCESS or FAIL

Notes

This call is for information only and it does not require any particular action.

## 6.14 *syWritePrintData*

### Prototype

```
NQ_UINT32 syWritePrintData(
    SYPrinterHandle handle,
    NQ_UINT32 jobId,
    const NQ_BYTE* data,
    NQ_UINT32 count,
    void **rctx
);
```

### Description

Print some data

### Parameters

handle	IN
Printer handle	
jobId	IN
Job ID as returned by <i>syStartPrintJob</i> (see 6.10)	
data	IN
Data to print	
count	IN
Number of bytes in the data	
rctx	IN/OUT
Address of a pointer to DCE RPC response context. See below.	

### Return Value

Number of bytes written

### Notes

This call passes to spooler a portion of data to be written to the printer. The parameter '*rctx*' is intended to support a case when printer is not able to accept data. In this case, it should save response context and initiate response later when printer will become available. The exact usage of this parameter requires knowledge of NQ internals; therefore, it is not fully documented here. For more information apply to VS customer support.

When data was fully accepted by the printer or application does not need late response, it must write a NULL pointer by the address specified as '*rctx*'.

## 6.15 *syGetPrintJobById*

### Prototype

```
NQ_STATUS syGetPrintJobById(  
    SYPrinterHandle handle,  
    NQ_UINT32 jobId,  
    SYPrintJobInfo* info  
);
```

### Description

Find job ID by its position in the print queue

### Parameters

handle	IN
Printer handle	
jobId	IN
Job ID	
Info	OUT
Buffer for job information structure (see 5.2.4)	

### Return Value

SUCCESS or FAIL

### Notes

Job ID is the number returned by *syStartPrintJob()* (see 6.10).  
Job information structure (see 5.2.4) contains several pointers. Those pointers when set should reference a persistent data, meaning that it should remain unchanged until the next call to the same function for the same printer.

## 6.16 *syGetPrintJobIdByIndex*

### Prototype

```
NQ_UINT32 syGetPrintJobIdByIndex(  
    SYPrinterHandle handle,  
    NQ_INT jobIdx  
);
```

### Description

Find job ID by its position in the print queue

### Parameters

handle	IN
Printer handle	
jobIdx	IN
Job position in the print queue for the given printer	

### Return Value

Job ID or FAIL

### Notes

Job ID is the number returned by *syStartPrintJob()* (see 6.10).

### 6.17 *syGetPrintJobIndexById*

#### Prototype

```
NQ_INT syGetPrintJobIndexById(  
    SYPrinterHandle handle,  
    NQ_INT jobId  
);
```

#### Description

Find job position in the print queue by its ID

#### Parameters

handle	IN
Printer handle	
jobId	IN
Job ID	

#### Return Value

Job position in the print queue for the given printer or FAIL

#### Notes

Job ID is the number returned by *syStartPrintJob()* (see 6.10).

## 6.18 *syGetPrintForm*

### Prototype

```
NQ_STATUS syGetPrintForm(
    SYPrinterHandle handle,
    NQ_INT formIdx,
    SYPrintFormInfo* info
);
```

### Description

Get form information

### Parameters

handle	IN
Printer handle	
formIdx	IN
Form index	
Info	OUT
Buffer for form information structure (see 5.2.7)	

### Return Value

SUCCESS or FAIL

### Notes

Form index is a zero-base form enumeration number, so that logically first form has index of zero, the next one has index of one and so on. Form information structure (see 5.2.7) contains several pointers. Those pointers when set should reference a persistent data, meaning that it should remain unchanged until the next call to the same function for the same printer.

## 6.19 *syControlPrinter*

### Prototype

```
NQ_STATUS syControlPrinter(  
    SYPrinterHandle handle,  
    NQ_UINT32 command  
);
```

### Description

Perform a printer command

### Parameters

handle	IN
Printer handle	
command	IN
Printer command code. See 5.1.6 for possible codes.	

### Return Value

SUCCESS or FAIL

### Notes

This call allows to perform a control operation on a printer.



## 6.20 *syControlPrintJob*

### Prototype

```
NQ_STATUS syControlPrintJob(  
    SYPrinterHandle handle,  
    NQ_UINT32 jobId,  
    NQ_UINT32 command  
);
```

### Description

Perform a job command

### Parameters

handle	IN
Printer handle	
jobId	IN
Job ID	
command	IN
Job command code. See 5.1.7 for possible codes.	

### Return Value

SUCCESS or FAIL

### Notes

Job ID is the number returned by *syStartPrintJob()* (see 6.10).

## 6.21 *syInitPrinters*

### Prototype

```
NQ_BOOL syInitPrinters (  
    SYPrinterPrepareSendLateResponse  
    printerPrepareSendLateWriteResponse,  
    NQ_COUNT responseContextSize  
);
```

### Description

Initialize the printer module

### Parameters

<code>printerPrepareSendLateWriteResponse</code>	IN
Pointer to callback for sending delayed response	
<code>responseContextSize</code>	IN
Delayed response size to be allocated when needed	

### Return Value

SUCCESS or FAIL

### Notes

NQ Core responsibility to initialize the printer module with the callback of type:  
typedef void (\*SYPrinterPrepareSendLateResponse)(SYPrinterJob \*,  
NQ\_INT);

**6.22 *syShutdownPrinters***

## Prototype

```
void syShutdownPrinters(void);
```

## Description

Shutdown the printer module

## Parameters

None

## Return Value

None

## Notes

This call deallocated the delayed response buffers per job

## **7 Implementation Notes**

This section contains useful implementation notes which do not fall into other categories.

### **7.1 Using DevMode Structure**

This structure (see 5.2.1) is part of printer information and it provides hardware information about the printer. Most of the fields in this structure are optional. The only mandatory field of those that are designated by the *field's* member is *formName* while the minimal combination of flags in the *field's* member should be *SY\_DEVICEMODE\_FORMNAME* (see 5.1.1).

It is a good idea, however, to supply more information about the printer. For instance, some Windows versions know to show in the printer window those jobs that it did not start spooling down to the printer yet. To show job information properly, Windows calculates number of pages and job size using printer geometry values provided in the DevMode structure. When printer does not provide those values through the DevMode structure, Windows calculations may result in showing weird values in the job information line.

## 7.2 Using Security Descriptors

Security Descriptor (further referenced as “SD”) stands for an information structure defining access rights of different users.

Two types of SD are relevant for printing:

- Printer SD
- Job SD

The first one is set or modified through the *syPrinterSetSecurityDescriptor()* call (see 6.9) and it is withdrawn through the *syPrinterGetSecurityDescriptor()* call (see 6.8). Printer SD can be also withdrawn from the *SYPrinterInfo* structure (see 5.2.3) obtained by calling *syGetprinterInfo()* (see 6.5). Job SD is set by means of *syStartPrintJob()* call (see 6.10) and withdrawn through the *SYPrintJobInfo* structure (see 5.2.4) obtained through the *syGetPrintJobById()* call (see 6.15).

NQ Server supports default descriptors for both printers and job. When NQ Server starts up it calls *syPrinterSetSecurityDescriptor()* (see 6.9) with a default SD. This SD designates the following access rights:

- Any user is allowed to submit jobs
- Only administrators can control printer

When client submits a new job, NQ Server generates a default SD for it with the following rights:

- Any user is allowed view jobs
- Only the user that submitted this job or an administrator can control it: pause, resume, cancel.

A spooler implementation may apply two different SD policies. First, it may use default SDs as provided by NQ Server. Then, it is spoolers responsibility to keep the SDs, but it does not modify them.

Another option is to ignore default SDs and to maintain internally generated SDs. This can be done by using optional SD library – SDLIB (currently not supplied) (see [4] ).

## 8 Sample Spooler

YNQ Software is shipped with a sample implementation of the Printing API. This implementation is for no means a real spooler but rather a basic simulation engine.

It features the following:

- Simulation of up to four printers is supported. Each printer should have a respective share with the name ending on the printer number (e.g., - “printer2”). This number starts from one (1) for the first printer.
- Printer information reports the following:
  - Two forms are reported: A4 and Letter.
  - Orientation – Portrait
  - Scale – 100
  - Print quality – 1200 dpi
  - Horizontal resolution – 1200 dpi.
  - The printer supports color

All other information fields contain random numbers.

- Sample implementation simulates HP LaserJet 4050 Series PCL6 driver for Windows XP/2000 only. All relevant files should reside in path [\\HData0a\\driver\\](#). A user should change this path to any other.
- Printing a document has no effect except for simulating printing process. A job status is reported as “Spooling” for the first 60 seconds and as “Printing” for the next 60 seconds. Then a job is removed.
- The progress of pages printed so far is always the number of pages in the print job and it does not change during job simulation.
- No printer-specific information is simulated.
- A job can be virtually cancelled, restarted and paused.

## 9 Direct Spooler

This is a basic spooler implementation (file syprintr.c for the Linux version of NQ ) serving the following purposes:

- It is a spooler implementation (or a bedrock for such implementation) for embedded device capable of plugging a voluntary printer.
- It is a more complete example of spooler implementation.
- It allows to simulate a real printer

Direct Spooler can control one or more printers plugged into USB slots. It is called “direct” because it prints jobs directly to the printer without saving it on intermediate media. Since Direct Spooler should be capable of supporting a voluntary printer it reports only the basic fields in printer information:

Direct Spooler has the following specification:

- Platforms – Linux only
- Works with a real printer as well as in a simulated mode
- Printer information reports the following:
  - Two forms are reported: A4 and Letter.
  - Orientation – unknown
  - Scale – unknown
  - Print quality – unknown
  - Horizontal resolution – unknown
  - Color support - unknown
- API restrictions – driver information is not implemented
- Number of printers supported in real mode – configurable
- Number of printers supported in simulated mode – one
- Number of concurrent jobs supported – configurable



### **9.1 Real Mode**

In the real mode Direct Spooler outputs received printing data directly to an attached printer. In case there is no printers plugged into the device adding printer as described in 4.1 will fail. Direct Spooler reports printer ID string, vendor and model codes from as received from the device itself.

## 9.2 *Simulated Mode*

The simulated mode allows to simulate one printer only of a specific model. In this mode there is no necessity to plug any real printer.

In this mode Direct Spooler reports hardcoded printer ID string, vendor and model codes. This information must be specified in the code of *syprintr.c* file. A number of printer models may be specified by adding information groups for each of them. However, only one of those groups should be compiled. Currently the only supported printer for simulated mode is Brother HL2030 series and it defined by default.

Adding a printer model involves the following steps.

- Add `#define SIMULATE_PRINTER_<YOURNAMEANDMODELHERE>`
- Add the following code lines:

```
#ifdef SIMULATE_PRINTER_<YOURNAMEANDMODELHERE>
#define SIMULATION_MODE
#define SHOW_PRINTING_PROCESS
#define SIMULATED_PRINTER_<ID>
#define SIMULATED_<PRINTER_VENDOR>
#define SIMULATED_<PRINTER_PRODUCT>
#endif
```

- Comment out all other “`#define SIMULATE_PRINTER_<NNN>`” pre-compiler blocks except the newly added one.
- Recompile.

Printing jobs in this mode is simulated with a delay of one second per a data portion which is usually 64K bytes long.

### **9.3 *Job management***

Direct Spooler is capable of accepting a number of simultaneous print jobs but only one of them is being spooled down to NQ while the spooling of subsequent job will start only when the current job completes or is cancelled. The jobs are printed in the order they arrived from clients. Since Direct Spooler transfers job data directly to the printer the term “spooling” above means is equivalent to “printing”.

## **9.4 Configuration**

Two Direct Spooler parameters can be configured in compile time.

- MAX\_PRINTERS – max number of simultaneously connected printers
- MAX\_JOBS – max number of concurrently serviced print jobs.