

[MS-RPRN]: Print System Remote Protocol Specification

Intellectual Property Rights Notice for Protocol Documentation

- This protocol documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the protocols, and may distribute portions of it in your implementations of the protocols or your documentation as necessary to properly document the implementation. This permission also applies to any documents that are referenced in the protocol documentation.
- Microsoft does not claim any trade secret rights in this documentation.
- Microsoft has patents that may cover your implementations of the protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. If you are interested in obtaining a patent license, please contact protocol@microsoft.com.
- The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

This protocol documentation is intended for use in conjunction with publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

A protocol specification does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them.

Revision Summary

Date	Revision History	Revision Class	Comments
02/22/2007	0.01		MCPPE Milestone 3 Initial Availability
06/01/2007	1.0	Major	Updated and revised the technical content.
07/03/2007	2.0	Major	Editorial changes, plus added a new section for a wsmon monitor module for the WSD_BACKUP_PORT_DATA packet structure.
07/20/2007	2.0.1	Editorial	Revised and edited the technical content.

Date	Revision History	Revision Class	Comments
08/10/2007	2.0.2	Editorial	Revised and edited the technical content.
09/28/2007	2.1	Minor	Updated the technical content.
10/23/2007	2.2	Minor	Updated the technical content.
11/30/2007	2.2.1	Editorial	Revised and edited the technical content.
01/25/2008	2.3	Minor	Updated the technical content.

Table of Contents

1	Introduction	10
1.1	Glossary	10
1.2	References	12
1.2.1	Normative References	12
1.2.2	Informative References.....	13
1.3	Protocol Overview (Synopsis).....	14
1.3.1	Management of the Print System	15
1.3.2	Communication of Print Job Data	16
1.3.3	Notification of Print System Changes	17
1.4	Relationship to Other Protocols.....	19
1.5	Prerequisites/Preconditions.....	19
1.6	Applicability Statement	19
1.7	Versioning and Capability Negotiation.....	19
1.8	Vendor-Extensible Members.....	20
1.9	Standards Assignments.....	20
2	Messages	21
2.1	Transport.....	21
2.2	Common Data Types	21
2.2.1	IDL Data Types	23
2.2.1.1	Common IDL Data Types.....	23
2.2.1.1.1	GDI_HANDLE	23
2.2.1.1.2	LANGID.....	23
2.2.1.1.3	PRINTER_HANDLE.....	24
2.2.1.1.4	RECTL.....	24
2.2.1.1.5	SIZE	24
2.2.1.1.6	STRING_HANDLE	25
2.2.1.1.7	UNIVERSAL_FONT_ID.....	25
2.2.1.2	Containers	26
2.2.1.2.1	DEVMODE_CONTAINER.....	26
2.2.1.2.2	DOC_INFO_CONTAINER.....	26
2.2.1.2.3	DRIVER_CONTAINER.....	26
2.2.1.2.4	FORM_CONTAINER.....	27
2.2.1.2.5	JOB_CONTAINER	28
2.2.1.2.6	MONITOR_CONTAINER	29
2.2.1.2.7	PORT_CONTAINER	29
2.2.1.2.8	PORT_VAR_CONTAINER.....	30
2.2.1.2.9	PRINTER_CONTAINER	31
2.2.1.2.10	RPC_BIDI_REQUEST_CONTAINER	32
2.2.1.2.11	RPC_BIDI_RESPONSE_CONTAINER	33
2.2.1.2.12	RPC_BINARY_CONTAINER.....	33
2.2.1.2.13	SECURITY_CONTAINER.....	33
2.2.1.2.14	SPLCLIENT_CONTAINER	34
2.2.1.2.15	STRING_CONTAINER.....	34
2.2.1.2.16	SYSTEMTIME_CONTAINER.....	35
2.2.1.3	Members in INFO Structures	35
2.2.1.3.1	DRIVER_INFO and RPC_DRIVER_INFO Members.....	35
2.2.1.3.2	FORM_INFO and RPC_FORM_INFO Members	37
2.2.1.3.3	JOB_INFO Members	37
2.2.1.3.4	MONITOR_INFO Members	39
2.2.1.3.5	PORT_INFO Members	39
2.2.1.3.6	PRINTER_INFO Members.....	39

2.2.1.3.7	SPLCLIENT_INFO Members.....	39
2.2.1.4	DOC_INFO_1.....	40
2.2.1.5	DRIVER_INFO.....	40
2.2.1.5.1	DRIVER_INFO_1.....	40
2.2.1.5.2	DRIVER_INFO_2.....	40
2.2.1.5.3	RPC_DRIVER_INFO_3.....	41
2.2.1.5.4	RPC_DRIVER_INFO_4.....	41
2.2.1.5.5	RPC_DRIVER_INFO_6.....	42
2.2.1.5.6	RPC_DRIVER_INFO_8.....	42
2.2.1.6	FORM_INFO.....	44
2.2.1.6.1	FORM_INFO_1.....	44
2.2.1.6.2	RPC_FORM_INFO_2.....	44
2.2.1.7	JOB_INFO.....	45
2.2.1.7.1	JOB_INFO_1.....	45
2.2.1.7.2	JOB_INFO_2.....	46
2.2.1.7.3	JOB_INFO_3.....	46
2.2.1.7.4	JOB_INFO_4.....	47
2.2.1.8	MONITOR_INFO.....	47
2.2.1.8.1	MONITOR_INFO_1.....	47
2.2.1.8.2	MONITOR_INFO_2.....	47
2.2.1.9	PORT_INFO.....	48
2.2.1.9.1	PORT_INFO_1.....	48
2.2.1.9.2	PORT_INFO_2.....	48
2.2.1.9.3	PORT_INFO_3.....	49
2.2.1.9.4	PORT_INFO_FF.....	50
2.2.1.10	PRINTER_INFO.....	51
2.2.1.10.1	PRINTER_INFO_STRESS.....	51
2.2.1.10.2	PRINTER_INFO_1.....	53
2.2.1.10.3	PRINTER_INFO_2.....	55
2.2.1.10.4	PRINTER_INFO_3.....	56
2.2.1.10.5	PRINTER_INFO_4.....	57
2.2.1.10.6	PRINTER_INFO_5.....	57
2.2.1.10.7	PRINTER_INFO_6.....	57
2.2.1.10.8	PRINTER_INFO_7.....	58
2.2.1.10.9	PRINTER_INFO_8.....	59
2.2.1.10.10	PRINTER_INFO_9.....	59
2.2.1.11	SPLCLIENT_INFO.....	59
2.2.1.11.1	SPLCLIENT_INFO_1.....	59
2.2.1.11.2	SPLCLIENT_INFO_3.....	60
2.2.1.12	Bidirectional Communication Data.....	60
2.2.1.12.1	RPC_BIDI_REQUEST_DATA.....	60
2.2.1.12.2	RPC_BIDI_RESPONSE_DATA.....	61
2.2.1.12.3	RPC_BIDI_DATA.....	61
2.2.1.13	Printer Notification Data.....	63
2.2.1.13.1	RPC_V2_NOTIFY_OPTIONS.....	63
2.2.1.13.2	RPC_V2_NOTIFY_OPTIONS_TYPE.....	63
2.2.1.13.3	RPC_V2_NOTIFY_INFO.....	64
2.2.1.13.4	RPC_V2_NOTIFY_INFO_DATA.....	65
2.2.1.13.5	RPC_V2_NOTIFY_INFO_DATA_DATA.....	66
2.2.1.13.6	RPC_V2_UREPLY_PRINTER.....	67
2.2.2	Custom-Marshaled Data Types.....	67
2.2.2.1	DEVMODE.....	68
2.2.2.1.1	PostScript Driver Extra Data.....	83
2.2.2.1.2	Generic Driver Extra Data.....	83
2.2.2.1.3	OEM Driver Extra Data.....	84

2.2.2.1.4	Print Ticket Driver Extra Data	84
2.2.2.2	Members in Custom-Marshaled INFO structures	84
2.2.2.3	ADDJOB_INFO_1	84
2.2.2.4	DATATYPES_INFO_1	85
2.2.2.5	_DRIVER_INFO	86
2.2.2.5.1	_DRIVER_INFO_1	86
2.2.2.5.2	_DRIVER_INFO_2	87
2.2.2.5.3	_DRIVER_INFO_3	89
2.2.2.5.4	_DRIVER_INFO_4	91
2.2.2.5.5	_DRIVER_INFO_5	93
2.2.2.5.6	_DRIVER_INFO_6	95
2.2.2.5.7	_DRIVER_INFO_7	98
2.2.2.5.8	_DRIVER_INFO_8	100
2.2.2.5.9	_DRIVER_INFO_101	104
2.2.2.5.10	_DRIVER_FILE_INFO	107
2.2.2.6	_FORM_INFO	108
2.2.2.6.1	_FORM_INFO_1	108
2.2.2.6.2	_FORM_INFO_2	109
2.2.2.7	_JOB_INFO	112
2.2.2.7.1	_JOB_INFO_1	112
2.2.2.7.2	_JOB_INFO_2	114
2.2.2.7.3	_JOB_INFO_3	117
2.2.2.7.4	_JOB_INFO_4	118
2.2.2.8	_MONITOR_INFO	121
2.2.2.8.1	_MONITOR_INFO_1	121
2.2.2.8.2	_MONITOR_INFO_2	122
2.2.2.9	_PORT_INFO	123
2.2.2.9.1	_PORT_INFO_1	123
2.2.2.9.2	_PORT_INFO_2	124
2.2.2.10	_PRINTER_INFO	126
2.2.2.10.1	_PRINTER_INFO_STRESS	126
2.2.2.10.2	_PRINTER_INFO_1	128
2.2.2.10.3	_PRINTER_INFO_2	130
2.2.2.10.4	_PRINTER_INFO_3	133
2.2.2.10.5	_PRINTER_INFO_4	134
2.2.2.10.6	_PRINTER_INFO_5	135
2.2.2.10.7	_PRINTER_INFO_6	136
2.2.2.10.8	_PRINTER_INFO_7	137
2.2.2.10.9	_PRINTER_INFO_8	138
2.2.2.10.10	_PRINTER_INFO_9	138
2.2.2.11	PRINTPROCESSOR_INFO_1	139
2.2.2.12	PRINTER_ENUM_VALUES	140
2.2.3	Constants	142
2.2.3.1	Status and Attribute Values	142
2.2.3.2	Flags Values	145
2.2.3.3	Printer Change Values	146
2.2.3.4	Access Values	148
2.2.3.5	Printer Notification Values	149
2.2.3.6	Job Notification Values	152
2.2.3.7	Change Notification Flag Values	154
2.2.3.8	Server Handle Key Values	155
2.2.3.9	Registry Type Values	155
2.2.3.10	BIDI_TYPE Enumeration	156
2.2.4	Rules for Members	157
2.2.4.1	Server Names	157

2.2.4.2	Printer Names.....	157
2.2.4.3	Access Values.....	159
2.2.4.4	Job Control Values.....	159
2.2.4.5	Environment Names	159
2.2.4.6	Driver Names	159
2.2.4.7	Path Names.....	159
2.2.4.8	Print Processor Names	160
2.2.4.9	Registry Type Values	160
2.2.4.10	Printer Change Values.....	160
2.2.4.11	Form Names.....	160
2.2.4.12	Monitor Names	160
2.2.4.13	Port Names	160
2.2.4.14	Print Provider Names	160
2.2.4.15	Datatype Names	160
2.2.4.16	Value Names	160
2.2.4.17	Key Names.....	160
2.2.4.18	User Names.....	161
3	Protocol Details	162
3.1	Server Details.....	162
3.1.1	Abstract Data Model.....	162
3.1.2	Timers	163
3.1.3	Initialization.....	163
3.1.4	Message Processing Events and Sequencing Rules	164
3.1.4.1	Commonly Used Parameters	174
3.1.4.1.1	Printer Name Parameters	174
3.1.4.1.2	Print Server Name Parameters.....	174
3.1.4.1.3	Datatype Name Parameters.....	175
3.1.4.1.4	Environment Name Parameters.....	175
3.1.4.1.5	INFO Structures Query Parameters.....	175
3.1.4.1.6	String Query Parameters.....	177
3.1.4.1.7	Dynamically Typed Query Parameters	178
3.1.4.1.8	PRINTER_ENUM_VALUES Structures Query Parameters	179
3.1.4.1.9	CONTAINER Parameters.....	179
3.1.4.1.9.1	DEVMODE_CONTAINER Parameters	180
3.1.4.1.9.2	DOC_INFO_CONTAINER Parameters.....	180
3.1.4.1.9.3	DRIVER_CONTAINER Parameters.....	180
3.1.4.1.9.4	FORM_CONTAINER Parameters.....	181
3.1.4.1.9.5	PORT_CONTAINER Parameters	181
3.1.4.1.9.6	PRINTER_CONTAINER Parameters	181
3.1.4.1.9.7	SECURITY_CONTAINER Parameters.....	182
3.1.4.1.9.8	SPLCLIENT_CONTAINER Parameters	182
3.1.4.1.10	PRINTER_HANDLE Parameters.....	182
3.1.4.1.11	Standard Parameter Validation.....	183
3.1.4.2	Printer Management and Discovery Methods.....	183
3.1.4.2.1	RpcEnumPrinters (Opnum 0)	186
3.1.4.2.2	RpcOpenPrinter (Opnum 1)	188
3.1.4.2.3	RpcAddPrinter (Opnum 5)	189
3.1.4.2.4	RpcDeletePrinter (Opnum 6).....	191
3.1.4.2.5	RpcSetPrinter (Opnum 7).....	192
3.1.4.2.6	RpcGetPrinter (Opnum 8).....	194
3.1.4.2.7	RpcGetPrinterData (Opnum 26)	196
3.1.4.2.8	RpcSetPrinterData (Opnum 27).....	197
3.1.4.2.9	RpcClosePrinter (Opnum 29)	198
3.1.4.2.10	RpcCreatePrinterIC (Opnum 40).....	199

3.1.4.2.11	RpcPlayGdiScriptOnPrinterIC (Opnum 41)	200
3.1.4.2.12	RpcDeletePrinterIC (Opnum 42)	201
3.1.4.2.13	RpcResetPrinter (Opnum 52)	202
3.1.4.2.14	RpcOpenPrinterEx (Opnum 69)	203
3.1.4.2.15	RpcAddPrinterEx (Opnum 70)	204
3.1.4.2.16	RpcEnumPrinterData (Opnum 72)	207
3.1.4.2.17	RpcDeletePrinterData (Opnum 73)	208
3.1.4.2.18	RpcSetPrinterDataEx (Opnum 77)	209
3.1.4.2.19	RpcGetPrinterDataEx (Opnum 78)	210
3.1.4.2.20	RpcEnumPrinterDataEx (Opnum 79)	212
3.1.4.2.21	RpcEnumPrinterKey (Opnum 80)	213
3.1.4.2.22	RpcDeletePrinterDataEx (Opnum 81)	214
3.1.4.2.23	RpcDeletePrinterKey (Opnum 82)	215
3.1.4.2.24	RpcAddPerMachineConnection (Opnum 85)	215
3.1.4.2.25	RpcDeletePerMachineConnection (Opnum 86)	216
3.1.4.2.26	RpcEnumPerMachineConnections (Opnum 87)	217
3.1.4.2.27	RpcSendRecvBidiData (Opnum 97)	218
3.1.4.3	Job Management Methods	220
3.1.4.3.1	RpcSetJob (Opnum 2)	220
3.1.4.3.2	RpcGetJob (Opnum 3)	222
3.1.4.3.3	RpcEnumJobs (Opnum 4)	223
3.1.4.3.4	RpcAddJob (Opnum 24)	225
3.1.4.3.5	RpcScheduleJob (Opnum 25)	226
3.1.4.4	Printer Driver Management Methods	227
3.1.4.4.1	RpcAddPrinterDriver (Opnum 9)	228
3.1.4.4.2	RpcEnumPrinterDrivers (Opnum 10)	229
3.1.4.4.3	RpcGetPrinterDriver (Opnum 11)	230
3.1.4.4.4	RpcGetPrinterDriverDirectory (Opnum 12)	232
3.1.4.4.5	RpcDeletePrinterDriver (Opnum 13)	233
3.1.4.4.6	RpcGetPrinterDriver2 (Opnum 53)	234
3.1.4.4.7	RpcDeletePrinterDriverEx (Opnum 84)	235
3.1.4.4.8	RpcAddPrinterDriverEx (Opnum 89)	237
3.1.4.5	Form Management Methods	240
3.1.4.5.1	RpcAddForm (Opnum 30)	240
3.1.4.5.2	RpcDeleteForm (Opnum 31)	241
3.1.4.5.3	RpcGetForm (Opnum 32)	242
3.1.4.5.4	RpcSetForm (Opnum 33)	243
3.1.4.5.5	RpcEnumForms (Opnum 34)	244
3.1.4.6	Port Management Methods	245
3.1.4.6.1	RpcEnumPorts (Opnum 35)	245
3.1.4.6.2	RpcDeletePort (Opnum 39)	247
3.1.4.6.3	RpcAddPortEx (Opnum 61)	248
3.1.4.6.4	RpcSetPort (Opnum 71)	249
3.1.4.6.5	RpcXcvData (Opnum 88)	250
3.1.4.7	Port Monitor Management Methods	251
3.1.4.7.1	RpcEnumMonitors (Opnum 36)	252
3.1.4.7.2	RpcAddMonitor (Opnum 46)	253
3.1.4.7.3	RpcDeleteMonitor (Opnum 47)	254
3.1.4.8	Print Processor Management Methods	255
3.1.4.8.1	RpcAddPrintProcessor (Opnum 14)	255
3.1.4.8.2	RpcEnumPrintProcessors (Opnum 15)	256
3.1.4.8.3	RpcGetPrintProcessorDirectory (Opnum 16)	258
3.1.4.8.4	RpcDeletePrintProcessor (Opnum 48)	259
3.1.4.8.5	RpcEnumPrintProcessorDatatypes (Opnum 51)	260
3.1.4.9	Document Printing Methods	261

3.1.4.9.1	RpcStartDocPrinter (Opnum 17)	261
3.1.4.9.2	RpcStartPagePrinter (Opnum 18)	262
3.1.4.9.3	RpcWritePrinter (Opnum 19)	263
3.1.4.9.4	RpcEndPagePrinter (Opnum 20)	264
3.1.4.9.5	RpcAbortPrinter (Opnum 21)	265
3.1.4.9.6	RpcReadPrinter (Opnum 22)	266
3.1.4.9.7	RpcEndDocPrinter (Opnum 23)	267
3.1.4.9.8	RpcFlushPrinter (Opnum 96)	268
3.1.4.10	Notification Methods	269
3.1.4.10.1	RpcWaitForPrinterChange (Opnum 28)	270
3.1.4.10.2	RpcFindClosePrinterChangeNotification (Opnum 56)	271
3.1.4.10.3	RpcRemoteFindFirstPrinterChangeNotification (Opnum 62)	271
3.1.4.10.4	RpcRemoteFindFirstPrinterChangeNotificationEx (Opnum 65)	273
3.1.4.10.5	RpcRouterRefreshPrinterChangeNotification (Opnum 67)	274
3.1.4.11	Monitor Module Methods	275
3.1.4.11.1	Localmon	276
3.1.4.11.2	Lprmon	276
3.1.4.11.3	Tcpmon	276
3.1.4.11.3.1	Structures	276
3.1.4.11.3.1.1	PORT_DATA_1	276
3.1.4.11.3.1.2	DELETE_PORT_DATA_1	276
3.1.4.11.3.1.3	CONFIG_INFO_DATA_1	276
3.1.4.11.3.1.4	PORT_DATA_LIST_1	276
3.1.4.11.3.1.5	PORT_DATA_2	276
3.1.4.11.3.2	Command Value Table	276
3.1.4.11.4	Wsdmon	276
3.1.4.11.4.1	Structures	276
3.1.4.11.4.1.1	WSD_DRIVER_DATA	276
3.1.4.11.4.1.2	WSD_BACKUP_PORT_DATA	277
3.1.4.11.4.2	Command Value Table	277
3.1.5	Timer Events	277
3.1.6	Other Local Events	277
3.2	Client Details	277
3.2.1	Abstract Data Model	277
3.2.2	Timers	277
3.2.3	Initialization	277
3.2.4	Message Processing Events and Sequencing Rules	278
3.2.4.1	Client-Side Notification-Processing Methods	278
3.2.4.1.1	RpcReplyOpenPrinter (Opnum 58)	278
3.2.4.1.2	RpcRouterReplyPrinter (Opnum 59)	279
3.2.4.1.3	RpcReplyClosePrinter (Opnum 60)	280
3.2.4.1.4	RpcRouterReplyPrinterEx (Opnum 66)	281
3.2.4.2	Client Interaction with the Print Server	282
3.2.4.2.1	Printing a Document Using RpcStartDocPrinter	282
3.2.4.2.2	Printing a Document Using RpcAddJob	282
3.2.4.2.3	Enumerating Printers on a Print Server	283
3.2.4.2.4	Enumerating Jobs on a Printer	283
3.2.4.2.5	Receiving Notifications from a Print Server	284
3.2.4.2.6	Announcing Shared Printers to Print Servers	285
3.2.4.2.7	Adding a Printer to a Print Server	285
3.2.5	Timer Events	286
3.2.6	Other Local Events	286
4	Protocol Examples	287
4.1	Adding a Printer to a Server	287

4.2	Adding a Printer Driver to a Server	287
4.3	Enumerating and Managing Printers	288
4.4	Printing a Job, Job Scheduling, and Setting Job Information	289
4.5	Enumerating Jobs and Modifying Job Settings.....	289
4.6	Receiving Notifications on Printing Events	290
5	Security	292
5.1	Security Considerations for Implementers	292
5.2	Index of Security Parameters	292
6	Appendix A: Full IDL	293
7	Appendix B: Windows Behavior	319
7.1	Version-Specific Additions.....	373
7.1.1	New in Windows Vista SP1 and Windows Server 2008	373
7.1.2	New in Windows Vista and Windows Server 2008	373
7.1.3	New in Windows XP SP1	374
7.1.4	New in Windows XP and Windows Server 2003.....	374
7.1.5	New in Windows 2000	374
7.1.6	New in Windows NT 4.0	375
7.1.7	New in Windows NT 3.51	376
7.1.8	New in Windows NT 3.5	376
7.1.9	Irregular Version-Specific Behavior.....	379
8	Index.....	380

1 Introduction

This document is a specification of the Print System Remote Protocol, a Microsoft proprietary interface. This protocol supports printing and spooling operations that are synchronous between client and server.

An enhanced replacement for this protocol is specified in [\[MS-PAR\]](#).

The Print System Remote Protocol defines the communication of **print job** processing and **print system** management between a **print client** and any **print server**. It is built on the **Remote Procedure Call (RPC)** Protocol.

1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

- Access Control Entry (ACE)**
- Active Directory (AD)**
- Authentication**
- Big-Endian**
- Binary Large Object (BLOB)**
- Color Profile**
- Device**
- Device Driver**
- Directory Service (DS)**
- Discretionary Access Control List (DACL)**
- Domain**
- Driver Package**
- Driver Store**
- Endpoint**
- Enhanced Metafile Format (EMF)**
- Enhanced Metafile (EMF) Spool Format**
- Globally Unique Identifier (GUID)**
- GUIDString**
- INF file**
- Interface Definition Language (IDL)**
- Little-Endian**
- Marshaling**
- Network Data Representation (NDR)**
- Opnum**
- Page Description Language (PDL)**
- PostScript**
- Principal**
- Print Client**
- Print Job**
- Print Queue**
- Print Server**
- Print System**
- Printer Control Language (PCL)**
- Printer Form**
- Registry**
- Remote Procedure Call (RPC)**
- RPC Context Handle**
- RPC Endpoint**
- RPC Engine**

RPC Transfer Syntax
RPC Transport
Security Identifier (SID)
Security Provider
Spool File
System Access Control List (SACL)
Unicode
Universal Naming Convention (UNC)
Universal Serial Bus (USB)
Universally Unique Identifier (UUID)
UTC
UTF-16LE
Well-Known Endpoint

The following terms are specific to this document:

Access Level: The type of access the client requests for an object, such as read access, write access, or administrative access.

Bidi, Bidirectional: The ability to move, transfer, or transmit in two directions.

Color Matching: The conversion of a color, sent from its original color space, to its visually closest color in the destination color space. See also **Image Color Management**.

Core Printer Driver: A **printer driver** that other **printer drivers** depend on. In Windows, this term includes the **Unidrv** and **Pscript5 printer drivers**.

Dithering: A form of digital **halftoning**.

Graphics Device Interface (GDI): A Windows API, which is supported on 16-bit and 32-bit versions of Windows, for performing graphics operations and image manipulation on logical graphics objects.

Halftoning: The process of converting grayscale or continuous-tone graphics or images to a representation with a discrete number of gray or tone levels.

Image Color Management (ICM): Technology that ensures that a color image, graphic, or text object is rendered as close as possible to its original intent on any **device**, despite differences in imaging technologies and color capabilities among **devices**.

Information Context: A special-purpose printer object that can only be used to obtain information about fonts that are supported by a printer.

Language Monitor: An executable object that provides a communications path between a **print queue** and a printer's **port monitor**. **Language monitors** add control information to the data stream, such as commands defined by a **Page Description Language (PDL)**. They are optional, and are only associated with a particular type of printer if specified in the printer's **INF file**.

Localmon: The that manages local serial ("COM") and parallel ("LPT") **ports** on a Windows machine.

Lprmon: The **port monitor module** that allows Windows **print servers** to send **print jobs** to machines that support UNIX **print server** functions.

Monitor Module: A monitor module is an executable object that provides a communication path between the **print system** and the printers on a server.

Multistring: An array of null-terminated, 16-bit **Unicode** strings (UTF-16LE-encoded), with one additional null after the final string, which is null-terminated in its own right. Thus, there are actually two nulls at the end of a **multistring** structure.

Multisiz: A data type defining a **multistring**.

N-Up Printing: The act of arranging multiple logical pages on a physical sheet of paper.

Plug-in: An executable module that can be loaded by the **print server** to perform specific functions.

Port: A logical name that represents a connection to a **device**. A port can represent a network address (for example, a TCP/IP address) or a local connection (for example, a **Universal Serial Bus (USB)** port).

Port Monitor: A port monitor is a **plug-in** that is responsible for communicating with a **device** that is connected to a **port**. A port monitor may interact with the **device** locally, remotely over a network, or through some other communication channel. The data that passes through a port monitor is in a form that can be understood by the destination **device**, such as **PDL**.

Port Monitor Module: A **monitor module** for a **port monitor**.

Print Processor: A **plug-in** that runs on the **print server** and processes **print job** data before it is sent to a print **device**.

Print Provider: A **plug-in** that runs on the **print server** and routes **print system** requests. Print providers are Windows-specific and not required by the protocol.

Print System Remote Protocol Stress Analysis: An optional diagnostic procedure that is used to analyze **print server** load, error counts, throughput, and other metrics.

Printer Driver: A software component that provides an interface between the operating system and the print **device**. It is responsible for processing the application data into a **PDL** that can be interpreted by the print **device**.

Printer Key: A string that uniquely identifies a path under the main **registry** key where printer configuration data is kept. Rules for printer key names are specified in section [2.2.4.17](#).

RAW Format: **PDL** data that can be sent to a **device** without further processing.

Tcpmon: The **port monitor module** that manages standard TCP/IP **ports** on a Windows machine. **Tcpmon** supports configuring a TCP/IP **port** and obtaining information about the **port** configuration.

White Point: The color value used as the reference to which the user adapts.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [\[RFC2119\]](#). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site,

<http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624>, as an additional source.

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, <http://www.opengroup.org/public/pubs/catalog/c706.htm>

[ICC] International Color Consortium, "Image Technology Colour Management - Architecture, Profile Format, and Data Structure", Specification ICC.1:2004-10, May 2006, http://www.color.org/icc_specs2.xalter

[MS-DTYP] Microsoft Corporation, "[Windows Data Types](#)", January 2007.

[MS-ERREF] Microsoft Corporation, "[Windows Error Codes](#)", January 2007.

[MS-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)", March 2007.

[MS-LCID] Microsoft Corporation, "[Windows Language Code Identifier \(LCID\) Reference](#)", July 2007.

[MS-RPCE] Microsoft Corporation, "[Remote Procedure Call Protocol Extensions](#)", January 2007.

[MS-SMB2] Microsoft Corporation, "[Server Message Block \(SMB\) Version 2.0 Protocol Specification](#)", July 2007.

[RFC1001] Network Working Group, "Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods", RFC 1001, March 1987, <http://www.ietf.org/rfc/rfc1001.txt>

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>

[RFC2616] Fielding, R., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, <http://www.ietf.org/rfc/rfc2616.txt>

[RFC2781] Hoffman, P. and Yergeau, F., "UTF-16, an encoding of ISO 10646", RFC 2781, February 2000, <http://www.ietf.org/rfc/rfc2781.txt>

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005, <http://www.ietf.org/rfc/rfc3986.txt>

[RFC4122] Leach, P., Mealling, M., and Salz, R., "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005, <http://www.ietf.org/rfc/rfc4122.txt>

1.2.2 Informative References

[DEVMODE] Microsoft Corporation, "DEVMODE", <http://msdn2.microsoft.com/en-us/library/ms535771.aspx>

[IEEE1284] Institute of Electrical and Electronics Engineers, "IEEE Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers—Description", IEEE Std 1284, 1994, http://standards.ieee.org/reading/ieee/std_public/description/busarch/1284-1994_desc.html

Note There is a charge to download the specification.

[MS-EMF] Microsoft Corporation, "[Enhanced Metafile Format Specification](#)", July 2007.

[MS-EMFSPool] Microsoft Corporation, "[Enhanced Metafile Spool Format Specification](#)", July 2007.

[MS-PAN] Microsoft Corporation, "[Print System Asynchronous Notification Protocol Specification](#)", January 2007.

[MS-PAR] Microsoft Corporation, "[Print System Asynchronous Remote Protocol Specification](#)", June 2007.

[MS-SPNG] Microsoft Corporation, "[Simple and Protected Generic Security Service Application Program Interface Negotiation Mechanism \(SPNEGO\) Protocol Extensions](#)", January 2007.

[MSDN-ADOVRVW] Microsoft Corporation, "Active Directory Schema Terminology", <http://msdn2.microsoft.com/en-us/library/ms675087.aspx>

[MSDN-MPD] Microsoft Corporation, "Microsoft Print Drivers", <http://msdn2.microsoft.com/en-us/library/aa907616.aspx>

[MSDN-SPOOL] Microsoft Corporation, "Print Spooler Components", <http://msdn2.microsoft.com/en-us/library/ms802196.aspx>

[MSDN-UINF] Microsoft Corporation, "Using INF Files", <http://msdn2.microsoft.com/en-us/library/Aa741213.aspx>

[MSDN-UDP] Microsoft Corporation, "Using Device Profiles with ICM2", <http://msdn2.microsoft.com/en-us/library/ms536847.aspx>

[MSDN-XMLP] Microsoft Corporation, Watson, B., "A First Look at APIs For Creating XML Paper Specification Documents", <http://msdn.microsoft.com//msdnmag/issues/06/01/xmlpaperspecification/default.aspx>

[RFC819] Su, Z.S. and Postel, J., "The Domain Naming Convention for Internet User Applications", RFC 819, August 1982, <http://www.ietf.org/rfc/rfc0819.txt>

[RFC1179] McLaughlin III, L., "Line Printer Daemon Protocol", RFC 1179, August 1990, <http://www.ietf.org/rfc/rfc1179.txt>

[WGFX] Yuan, F., "Windows Graphics Programming - Win32 GDI and DirectDraw", Prentice Hall PTR, 2000, ISBN: 0130869856.

If you have any trouble finding [WGFX], please check [here](#).

1.3 Protocol Overview (Synopsis)

The Print System Remote Protocol provides the following functions:

- Management of the print system of a print server from a client
- Communication of print job data from a client to a print server
- Notifications to the client of changes in the print server's print system

Server processing instructions are specified by the parameters that are used in the protocol methods. These parameters include:

- **Printer driver** configuration information.
- The **spool file** format for the print data that is sent by the client.
- The **access level** of the connection.
- The target **print queue** name for name-based methods.
- A handle to the target print queue for handle-based methods.

Status information is communicated back to the client in the return codes from calls that are made to the print server.

The following sections give an overview of these functions.

1.3.1 Management of the Print System

A client can use this protocol to perform remote management operations on a print server. With server access credentials, client applications can manipulate the print server state and print server components, such as printer driver configuration and print queue configuration, or add printer drivers and printers; they can monitor the print queue status; and they can perform general print server administration.

These operations are supported in the protocol by a set of [container](#) structures that are used by different print system components, specifically: [DRIVER CONTAINER](#), [FORM CONTAINER](#), [JOB CONTAINER](#), [PORT CONTAINER](#), [SECURITY CONTAINER](#), and [PRINTER CONTAINER](#). These print system components are supported as specified in section [2.2.1](#).

To produce printed output that is the same, regardless of the configuration, the printer driver that is installed on the client computer must be identical to or compatible with the printer driver that is installed on the print server. This protocol provides the methods that the client can use after it connects to a printer on a print server to obtain the information about the printer driver that is associated with the printer. If necessary, the client computer can use this information to download the printer driver from the print server. For more information about printer drivers, see [\[MSDN-MPD\]](#).

The client can also use this protocol to obtain detailed information about the settings of the printer and the printer driver that are installed on the server. The client application can use this information to perform layout and to make device-specific choices about paper formats, resolution, and color handling. After the client connects to a printer, this protocol provides the methods that the client can use to query these settings.

The following diagram illustrates this interaction, using the scenario of adding a new printer as an example.

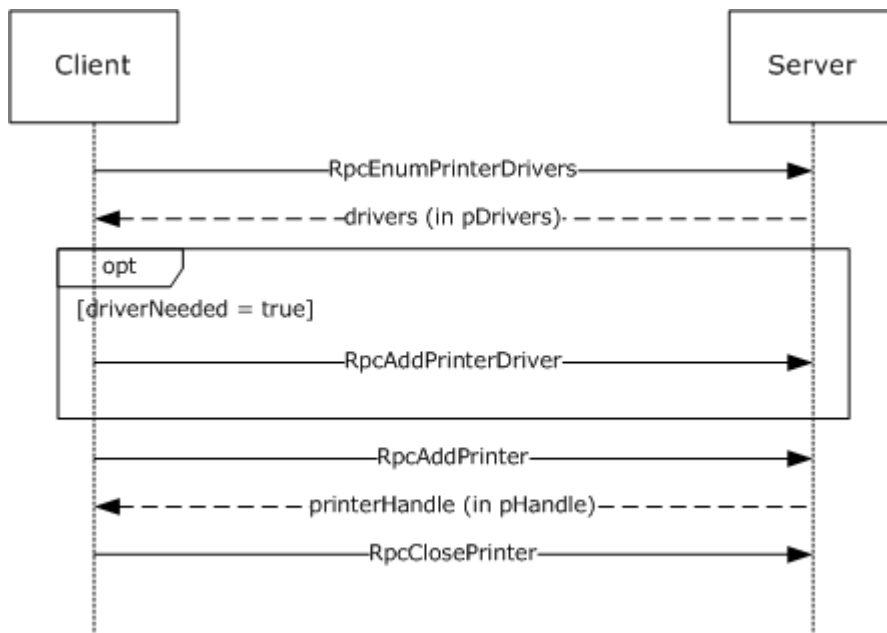


Figure 1: Adding a new printer

1.3.2 Communication of Print Job Data

Communication of print job data enables a client to print to **devices** that are hosted by print servers.

In one configuration, a client uses a printer driver that is installed on the client computer to convert a graphical representation of application content and layout into device-specific **Page Description Language (PDL)** data. It then sends the data, also called RAW data, to the print server using methods this protocol provides. The print server can temporarily store the RAW data from the client in a spool file, or it can print it immediately. As the print server sends the data to the target printer, the **print processor** on the print server that is associated with the target printer can post-process the RAW data in an implementation-specific way.

In another configuration, a client sends data to the print server in an intermediate format that contains graphics primitives and layout information in addition to processing instructions for the print server. The print server can temporarily store this intermediate data in a spool file, or it can print it immediately. As the data is sent to the printer, the **print processor** on the print server that is associated with the printer converts the data from the intermediate spool file to device-specific PDL data, typically by using the printer driver that is installed on the print server. [<1>](#)

The following diagram illustrates this interaction.

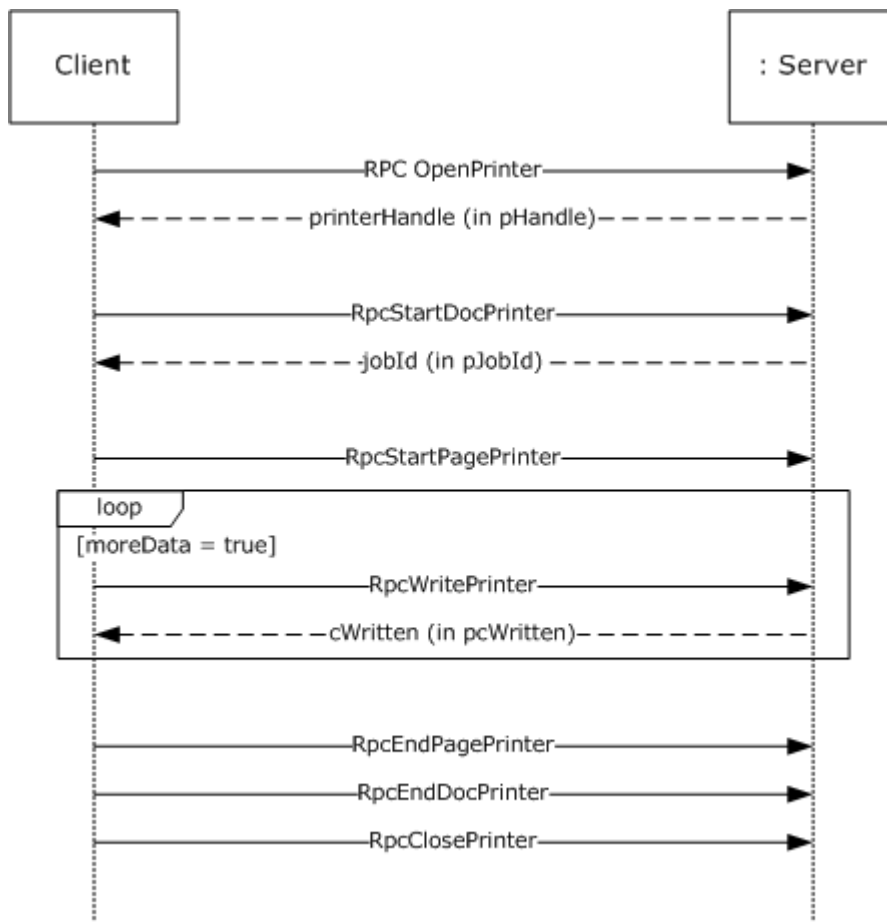


Figure 2: Communication of print job data

1.3.3 Notification of Print System Changes

This protocol also provides the methods that a client can use to register for incremental change notifications. These notifications enable the client application to maintain an accurate local view of the printer and printer driver settings by enabling the client application to synchronize the local view with the actual settings of those components on the print server, without having to repeatedly query the server for its complete configuration information.

For status updates, a client registers for notifications of state changes when it connects to a print server. The server creates a new Remote Procedure Call (RPC) connection in the reverse direction (back to the client), which is subsequently used to send notifications to the client. When the status of a server resource changes—such as a print queue goes online, goes offline, or enters an error state—the server sends a notification to the registered client.

Notifications include status changes of print server resources; for example, when a print queue goes online, goes offline, or enters an error state.

The following diagram illustrates this interaction. For more information, see section [3.2.4.2.5](#).

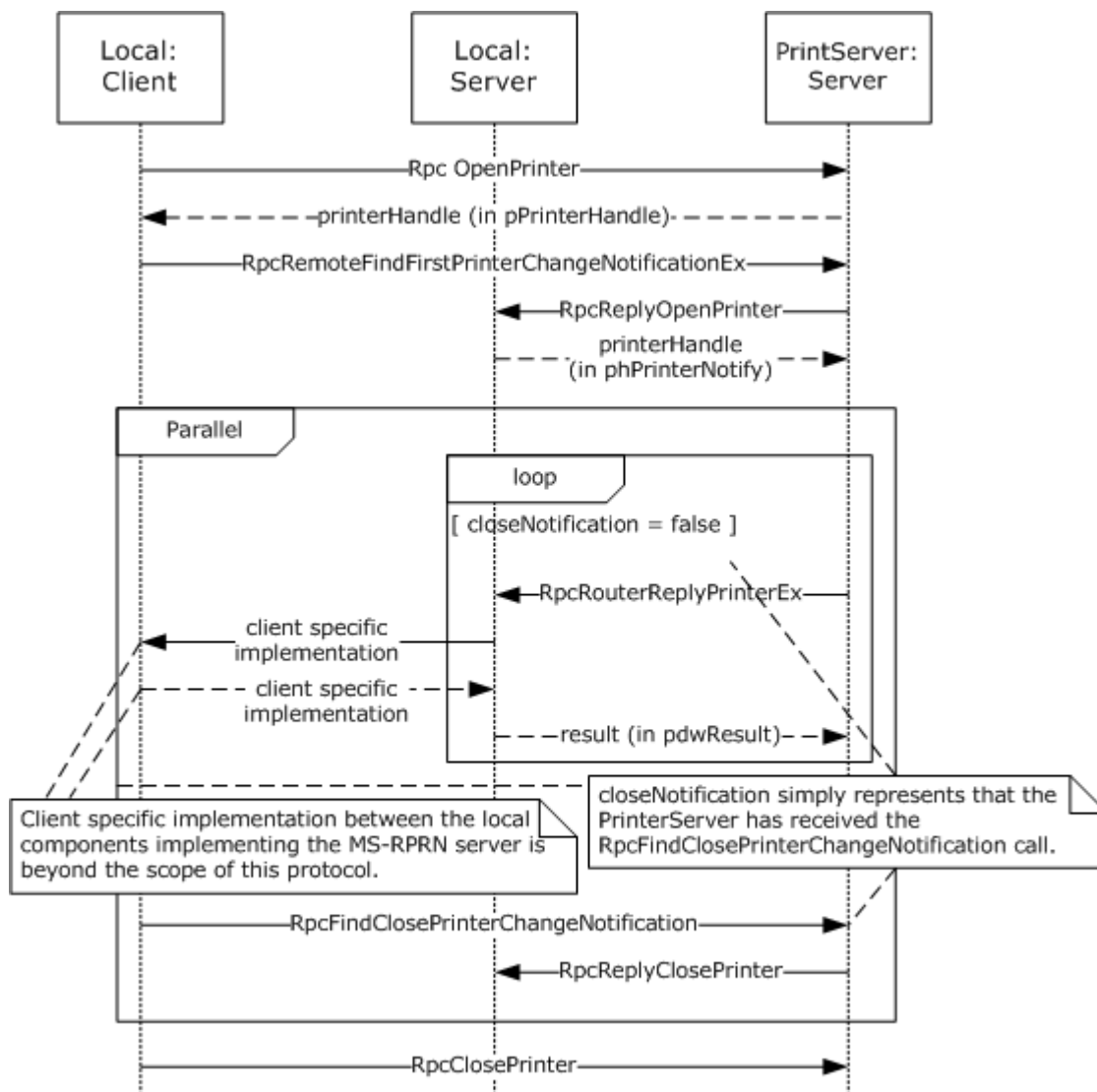


Figure 3: Notification of print system changes

In addition to composing and returning the notifications, the print server maintains a change identifier that it changes whenever the server-side printing configuration changes; for example, changes to user-configurable settings, print queue items, print job status, or to the printer driver would cause this identifier to change. The client can query this change identifier by using the [RpcGetPrinterData](#) method that is defined in this protocol and calling it with the **pValueName** parameter pointing to the string "ChangeID".

When a disconnected client reconnects to the print server, it can query the change identifier again, and if the change identifier is different from the one returned when it queried before it was disconnected, the client should retrieve the complete configuration information and update its view of the server configuration. The client retrieves the complete configuration using the functions for "[Management of the Print System](#)".

1.4 Relationship to Other Protocols

The Print System Remote Protocol is dependent on remote procedure call (RPC), as specified in [\[MS-RPCE\]](#).

This protocol does not specify methods for file transfer between client and server. The [Server Message Block \(SMB\) Version 2.0 Protocol](#), specified in [MS-SMB2], SHOULD be used for any file transfer between client and server, such as driver download operations.

The [Print System Asynchronous Remote Protocol](#), specified in [MS-PAR], uses many data structures and parameter definitions that are specified in this protocol.

The [Print System Asynchronous Notification Protocol](#), as described in [MS-PAN], is dependent on the Print System Remote Protocol.

1.5 Prerequisites/Preconditions

This protocol is an RPC interface and therefore has the prerequisites specified in [\[MS-RPCE\]](#) section 1.5 as being common to RPC interfaces.

It is assumed that a client of this protocol has obtained the name of a server that supports this protocol before it is invoked. For information on how a client does this, see [\[MSDN-MPD\]](#).

1.6 Applicability Statement

The Print System Remote Protocol is applicable only for printing operations between a system functioning as a client and a system functioning as a print server. This protocol scales from home use, in which a single printer is connected to a single computer; to office use, in which print-devices are shared between computers; to enterprise use, in which multiple print servers are employed in a cluster configuration, and the client configuration is managed by a directory access protocol. [<2>](#)

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Supported Transports:** The Print System Remote Protocol MUST use Remote Procedure Call (RPC) over named pipes only.
- **Protocol Versions:** The protocol version specified in the **Interface Definition Language (IDL)** file MUST be one. Versioning of data structures defined by the protocol is controlled using a level value in all [CONTAINER data structures](#). Typically, levels are sequential, and data structures identified by a later-version level, if extending an earlier level, are a superset of the data structure identified by the earlier level. Levels and containers are defined in section [2.2.1](#). The level value is also a parameter to some RPC methods. If a protocol method fails because the information level is not supported, the client SHOULD try with a lower level.
- **Security and Authentication Methods:** Versioning of security is handled by the underlying **RPC transport**; see [\[MS-RPCE\]](#) section 3.3.3.3 for more information.
- **Localization:** This protocol does not contain locale-specific information.
- **Return Values:** The methods that make up this RPC interface MUST return zero to indicate successful completion and nonzero values to indicate failure, except where specifically described. Unless otherwise specified, a server-side implementation of this protocol SHOULD choose any nonzero Win32 error value to signify an error condition, as described in section [1.8](#). Unless otherwise specified, the client side of the Print System Remote Protocol MUST NOT

interpret returned error codes. Unless otherwise specified, the client side of the Print System Remote Protocol MUST simply return error codes to the invoking application without taking any protocol action.

- **Capability Negotiation:** Functional negotiation is supported through the use of CONTAINER levels; see section [2.2.1](#). On connection to a server, the client requests a level. If the information level is a level supported by the server, the server MUST process the request. Otherwise, the server MUST return an error to the client, and the client SHOULD repeat the request with a lower level.

1.8 Vendor-Extensible Members

The methods defined in the Print System Remote Protocol specify either the DWORD or **HRESULT** data type for return values. DWORD return values are Win32 error codes taken from the Windows error number space specified in [\[MS-ERREF\]](#). Implementers MUST reuse those values with their indicated meanings. Choosing any other value runs the risk of collisions.

HRESULT method return values are used as defined in [\[MS-ERREF\]](#). Vendors are free to choose their own **HRESULT** values, but the C bit (0x20000000) MUST be set, indicating it is a customer code. [<3>](#)

Print server implementations must generate **Globally Unique Identifier (GUID)** strings for the purpose of identifying specific printers. The 128-bit value encoded by the GUID string SHOULD conform to the specification in [\[RFC4122\]](#) section 4.

1.9 Standards Assignments

The Print System Remote Protocol MUST use the following private assignments:

- RPC **Universally Unique Identifier (UUID)** of 12345678-1234-ABCD-EF00-0123456789AB. For information on transport details, see section [2.1](#).
- RPC **well-known endpoint** of **\pipe\spoolss**. For information on transport details, see section [2.1](#).

2 Messages

The following sections specify how Print System Remote Protocol messages are transported and common data types.

2.1 Transport

The Print System Remote Protocol MUST use the following transport for RPC sequences:

- RPC over named pipes, as specified in [\[MS-RPCE\]](#) section 2.1.1.2.

This protocol MUST use the following well-known endpoints:

- For RPC over named pipes, as specified in [\[MS-RPCE\]](#) section 2.1.1.2, the **endpoint** is `\pipe\spoolss`. This endpoint is used for RPC calls made from the client to the server. The client MUST use no **authentication**. The server MUST accept connections without authentication.

An endpoint with the same name MUST also be used for RPC calls made from the server back to the client, to send printer change notifications; those RPC calls for notifications are [RpcReplyOpenPrinter \(section 3.2.4.1.1\)](#), [RpcRouterReplyPrinter \(section 3.2.4.1.2\)](#), [RpcReplyClosePrinter \(section 3.2.4.1.3\)](#), and [RpcRouterReplyPrinterEx \(section 3.2.4.1.4\)](#). The client MUST accept connections without authentication from the server for these methods.<4>

2.2 Common Data Types

The Print System Remote Protocol MUST indicate to the RPC runtime that it is to support both the **Network Data Representation (NDR)** and **NDR64 RPC transfer syntaxes** and provide a negotiation mechanism for determining which transfer syntax will be used, as specified in [\[MS-RPCE\]](#) section 3.

This protocol MUST enable **the ms_union** extension as specified in [\[MS-RPCE\]](#) section 2.2.4.

The Print System Remote Protocol employs a combination of the following data representations:

- Interface Definition Language (IDL) data structures used with RPC methods, including structures used as containers for custom-**marshaled**, custom C data, are specified in section [2.2.1](#).
- Custom C data structures and their wire formats used within custom marshaled data streams are specified in section [2.2.2](#).

Unless noted otherwise, the following statements apply to this specification:

- All strings defined in this protocol MUST consist of characters encoded in **Unicode UTF-16LE** and MUST be null-terminated. Each UTF-16 codepoint in a string, including terminating null characters, MUST occupy 16 bits. Details are as specified in [\[RFC2781\]](#), section 2.1.
- A list of strings is referred to as a **multisiz** in this protocol. In a **multisiz**, the characters making up the string N+1 MUST directly follow the terminating null character of string N. The last string in a **multisiz** MUST be terminated by two null characters.
- All parameters (or members) specifying the number of characters in a string or **multisiz** specify the number of characters including terminating null characters.
- All constraints specifying the maximum number of characters in a string or **multisiz** specify the number of characters including terminating null characters.

- All parameters (or members) specifying the number of bytes in buffers containing a string or **multisiz** specify the number of bytes including terminating null characters.
- All data types made up of more than a single byte MUST be treated as specified in **little-endian** byte order.
- The term "null" means "a null pointer", and "zero" means the number 0.
- All parameters (or members) specifying the size of a buffer pointed to by another parameter (of member) MUST be zero if the pointer parameter (or member) is null.
- The term "**empty string**" means a string containing only the terminating null character.

This protocol introduces a variety of data types that bundle information about printers, printer drivers, print jobs, and **ports**. These data types are collectively referred to as "**INFO**" data types, and include [DRIVER_INFO_1](#), [PRINTER_INFO_1](#), [JOB_INFO_1](#), and [PORT_INFO_1](#). As data types were refined in the evolution of this protocol, new "INFO" data type versions have been introduced to represent extended or different bundles of information. The term "level" is used to differentiate between the different type versions, and the number of the level is reflected in the name of the data type, for example, [JOB_INFO_1](#), [JOB_INFO_2](#), [JOB_INFO_3](#).

To simplify method parameter lists and to increase robustness of RPC marshaling, the protocol introduces "[CONTAINER](#)" data types, which hold a level value along with a union of pointer values pointing to different "INFO" data type versions. For example, a [JOB_CONTAINER](#) contains a level value and a union of pointers to the different [JOB_INFO](#) data type versions, which are selected by the value of the level value.

Most of the "INFO" data types have an IDL form and a custom-marshaled form. IDL forms are typically used in conjunction with "CONTAINER" data types, as input parameters to methods that set values, such as [RpcSetPrinter](#); while custom-marshaled forms are typically used as output parameters to methods that get values, such as [RpcGetPrinter](#). The layout and order of members of IDL forms are typically the same as those of corresponding custom-marshaled forms, with the distinction that IDL forms use the type "[string] wchar_t *" to point to strings, while custom-marshaled forms use an offset relative to the start of the structure.

As an exception to the above rule, the layout of IDL-marshaled structures that contain pointers to **multisiz** data **differs** from the layout of custom-marshaled forms, in that IDL-marshaled structures need to define a **length** member for each IDL-marshaled member of type **pointer to multisiz**; the names of such IDL-marshaled members start with "RPC_".

To increase clarity, an underscore has been prepended to the names of all custom-marshaled structures, for which an IDL-marshaled form exists. For example, [_DRIVER_INFO_1](#) is the name of the custom-marshaled structure that corresponds to [DRIVER_INFO_1](#), the IDL-marshaled form.

When IDL-marshaled structures that contain pointer types to variable-length data without appropriate IDL attributes, such as "[string]" or "[size_is(...)]", are used as input arguments to methods, either directly or in CONTAINER structures, the pointers and variables to which they point cannot be marshaled by RPC, because RPC does not know the length of the data that is pointed to. Examples are the **pSecurityDescriptor** and **pDevMode** members of the [PRINTER_INFO](#) structures.

To address this problem, methods that specify such input arguments accept separate CONTAINER structures that pass in custom-marshaled or self-relative forms of the pointers and the variables that they reference. Examples of such methods are [RpcSetPrinter](#) and [RpcAddPrinterEx](#). Individual method sections specify how affected pointer members and CONTAINER structures MUST be treated.

2.2.1 IDL Data Types

In addition to the RPC base types and definitions specified in [\[C706\]](#) and [\[MS-DTYP\]](#), the Print System Remote Protocol defines data types in the following sections:

- [Common IDL Data Types \(section 2.2.1.1\)](#)
- [Containers \(section 2.2.1.2\)](#)
- [Members in INFO Structures \(section 2.2.1.3\)](#)
- **[DOC_INFO_1 \(section 2.2.1.4\)](#)**
- [DRIVER_INFO \(section 2.2.1.5\)](#)
- [FORM_INFO \(section 2.2.1.6\)](#)
- [JOB_INFO \(section 2.2.1.7\)](#)
- [MONITOR_INFO \(section 2.2.1.8\)](#)
- [PORT_INFO \(section 2.2.1.9\)](#)
- [PRINTER_INFO \(section 2.2.1.10\)](#)
- [SPLCLIENT_INFO \(section 2.2.1.11\)](#)
- [Bidirectional Communication Data \(section 2.2.1.12\)](#)
- [Printer Notification Data \(section 2.2.1.13\)](#)

2.2.1.1 Common IDL Data Types

2.2.1.1.1 GDI_HANDLE

The **GDI_HANDLE** serves as an **RPC context handle** for methods that specify a printer **information context** handle parameter. RPC context handles are specified in [\[C706\]](#) sections 2.3 and 6.1.6.

This type is declared as follows:

```
typedef [context_handle] void* GDI_HANDLE;
```

The **GDI_HANDLE** context handle is returned by [RpcCreatePrinterIC](#).

2.2.1.1.2 LANGID

The **LANGID** data type identifies the human language used for the user interface for printing. Details are specified in [\[MS-LCID\]](#).

This type is declared as follows:

```
typedef unsigned short LANGID;
```

2.2.1.1.3 PRINTER_HANDLE

The **PRINTER_HANDLE** serves as an RPC context handle for methods that specify a printer object handle parameter. RPC context handles are specified in [\[C706\]](#) sections 2.3 and 6.1.6.

This type is declared as follows:

```
typedef [context_handle] void* PRINTER_HANDLE;
```

The PRINTER_HANDLE context handle is returned by [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#) and [RpcOpenPrinterEx](#).

2.2.1.1.4 RECTL

The **RECTL** structure defines a rectangle on a form, with two (x,y) coordinates in thousandths of a millimeter units.

```
typedef struct {  
    long left;  
    long top;  
    long right;  
    long bottom;  
} RECTL;
```

left: The value of this member MUST specify the x-coordinate of the upper-left corner of the rectangle relative to the left edge of the form. This value MUST be an integer greater than or equal to 0, and it MUST be smaller than or equal to the width of the form.

top: The value of this member MUST specify the y-coordinate of the upper-left corner of the rectangle relative to the top edge of the form. This value MUST be an integer greater than or equal to 0, and it MUST be smaller than or equal to the height of the form.

right: The value of this member MUST specify the x-coordinate of the lower-right corner of the rectangle relative to the left edge of the form. This value MUST be greater than or equal to **left**.

bottom: The value of this member MUST specify the y-coordinate of the lower-right corner of the rectangle relative to the top edge of the form. This value MUST be greater than or equal to **top**.

2.2.1.1.5 SIZE

The **SIZE** structure defines the area of a form, with a width and height in thousandths-of-a-millimeter units.

```
typedef struct {
    long cx;
    long cy;
} SIZE;
```

cx: The value of this member MUST specify the width, and it MUST be an integer greater than or equal to 0.

cy: The value of this member MUST specify the height, and it MUST be an integer greater than or equal to 0.

2.2.1.1.6 STRING_HANDLE

The **STRING_HANDLE** serves as an RPC binding handle for methods that do not specify a **PRINTER_HANDLE** parameter. RPC binding handles are specified in [\[C706\]](#) section 2.3.

This type is declared as follows:

```
typedef [handle] wchar_t* STRING_HANDLE;
```

To build the binding handle for those methods, RPC requires a **protocol sequence**, a network address, and an endpoint. Both the protocol sequence and the endpoint are bound to the RPC interface; they MUST be named pipes and **\pipe\spoolss**, respectively. The network address MUST be defined by the printer or print server name. The printer name is usually in the form **\\server\printer** and the server MUST be used as the network address.

2.2.1.1.7 UNIVERSAL_FONT_ID

The **UNIVERSAL_FONT_ID** structure defines the identity of a font.

```
typedef struct {
    ULONG CheckSum;
    ULONG Index;
} UNIVERSAL_FONT_ID;
```

CheckSum: A 32-bit unsigned integer that is the implementation-specific checksum of the font.[<5>](#)

Index: A 32-bit unsigned integer that is an index associated with the font. The meaning of this field is determined by the type of font.

For additional information, see [\[MS-EMF\]](#) section 2.2.24.

2.2.1.2 Containers

2.2.1.2.1 DEVMODE_CONTAINER

The **DEVMODE_CONTAINER** structure defines a buffer size and pointer to a buffer that MUST contain a [DEVMODE](#) structure. DEVMODE data is used in the initialization of printer drivers. The custom-marshaled specification of DEVMODE is in section [2.2.2.1](#).

```
typedef struct _DEVMODE_CONTAINER {  
    DWORD cbBuf;  
    [size_is(cbBuf), unique] BYTE* pDevMode;  
} DEVMODE_CONTAINER;
```

cbBuf: This member MUST contain the size, in bytes, of the buffer pointed to by the **pDevMode** member.

pDevMode: This member MUST be either null or a non-null pointer to a DEVMODE structure, specified in section [2.2.2.1](#). A null value indicates that the initialization values of the default DEVMODE for the current printer driver SHOULD be used.

2.2.1.2.2 DOC_INFO_CONTAINER

The **DOC_INFO_CONTAINER** structure provides information about the document to be printed, using the [DOC_INFO_1](#) structure.

```
typedef struct _DOC_INFO_CONTAINER {  
    DWORD Level;  
    [switch_is(Level)] union {  
        [case(1)]  
        DOC_INFO_1* pDocInfo1;  
    } DocInfo;  
} DOC_INFO_CONTAINER;
```

Level: This member specifies the information level of the **DocInfo** member data. The value of this member MUST be set to 0x00000001.

DocInfo: This member MUST define document properties, using an information structure that MUST correspond to the value of the **Level** member.

pDocInfo1: This member MUST be a non-null pointer to a **DOC_INFO_1** structure that describes the document that will be printed. Details are specified in section [2.2.1.4](#).

2.2.1.2.3 DRIVER_CONTAINER

The **DRIVER_CONTAINER** structure provides information about printer drivers, using [DRIVER_INFO](#) structures. The **DriverInfo** member specifies the structure that defines the printer driver properties.

```
typedef struct _DRIVER_CONTAINER {  
    DWORD Level;  
    [switch_is(Level)] union {
```

```

[case(1)]
    DRIVER_INFO_1* Level1;
[case(2)]
    DRIVER_INFO_2* Level2;
[case(3)]
    RPC_DRIVER_INFO_3* Level3;
[case(4)]
    RPC_DRIVER_INFO_4* Level4;
[case(6)]
    RPC_DRIVER_INFO_6* Level6;
[case(8)]
    RPC_DRIVER_INFO_8* Level8;
} DriverInfo;
} DRIVER_CONTAINER;

```

Level: This member MUST specify the information level of the **DriverInfo** data. The value of this member MUST be in the range 0x00000001 to 0x00000004 inclusive, 0x00000006, or 0x00000008.

DriverInfo: This member MUST define printer driver properties, using an information structure that MUST correspond to the value of the **Level** member.

Level1: If the **Level** member is 1, this member MUST be a non-null pointer to a **DRIVER_INFO_1** structure providing printer driver information. For details, see section [2.2.1.5.1](#).

Level2: If the **Level** member is 2, this member MUST be a non-null pointer to a **DRIVER_INFO_2** structure providing printer driver information. For details, see section [2.2.1.5.2](#).

Level3: If the **Level** member is 3, this member MUST be a non-null pointer to an **RPC_DRIVER_INFO_3** structure providing printer driver information. For details, see section [2.2.1.5.3](#).

Level4: If the **Level** member is 4, this member MUST be a non-null pointer to an **RPC_DRIVER_INFO_4** structure providing printer driver information. For details, see section [2.2.1.5.4](#).

Level6: If the **Level** member is 6, this member MUST be a non-null pointer to an **RPC_DRIVER_INFO_6** structure providing printer driver information. For details, see section [2.2.1.5.5](#).

Level8: If the **Level** member is 8, this member MUST be a non-null pointer to an **RPC_DRIVER_INFO_8** structure providing printer driver information. For details, see section [2.2.1.5.6](#).

2.2.1.2.4 FORM_CONTAINER

The **FORM_CONTAINER** structure provides information about **printer forms**, using [FORM_INFO](#) structures. The **FormInfo** member specifies the structure that defines the printer form properties.

```

typedef struct _FORM_CONTAINER {
    DWORD Level;
    [switch_is(Level)] union {

```

```

    [case(1)]
        FORM_INFO_1* pFormInfo1;
    [case(2)]
        RPC_FORM_INFO_2* pFormInfo2;
    } FormInfo;
} FORM_CONTAINER;

```

Level: This member MUST specify the information level of the **FormInfo** data. The value of this member MUST be 0x00000001 or 0x00000002.

FormInfo: This member MUST define printer form properties, using an information structure that MUST correspond to the value of the **Level** member.

pFormInfo1: If the **Level** member is 0x00000001, this member MUST be a non-null pointer to a [FORM_INFO_1](#) structure, which provides information about a printer form. For details, see section [2.2.1.6.1](#).

pFormInfo2: If the **Level** member is 0x00000002, this member MUST be a non-null pointer to a [RPC FORM_INFO_2](#) structure, which provides information about a printer form. For details, see section [2.2.1.6.2](#).

2.2.1.2.5 JOB_CONTAINER

The **JOB_CONTAINER** structure provides information about print jobs, using [JOB_INFO](#) structures. The **JobInfo** member specifies the structure that defines the print job properties.

```

typedef struct _JOB_CONTAINER {
    DWORD Level;
    [switch_is(Level)] union {
        [case(1)]
            JOB_INFO_1* Level1;
        [case(2)]
            JOB_INFO_2* Level2;
        [case(3)]
            JOB_INFO_3* Level3;
        [case(4)]
            JOB_INFO_4* Level4;
    } JobInfo;
} JOB_CONTAINER;

```

Level: This member MUST specify the information level of the **JobInfo** data. The value of this member MUST be in the range 0x00000001 to 0x00000004 inclusive.

JobInfo: This member MUST define print job properties, using an information structure that MUST correspond to the value of the **Level** member.

Level1: If the **Level** member is 1, this member MUST be a non-null pointer to a [JOB_INFO_1](#) structure that provides print job information. For details, see section [2.2.1.7.1](#).

Level2: If the **Level** member is 2, this member MUST be a non-null pointer to a [JOB_INFO_2](#) structure that provides print job information. For details, see section [2.2.1.7.2](#).

Level3: If the **Level** member is 3, this member MUST be a non-null pointer to a [JOB_INFO_3](#) structure that provides print job information. For details, see section [2.2.1.7.3](#).

Level4: If the **Level** member is 4, this member MUST be a non-null pointer to a [JOB_INFO_4](#) structure that provides print job information. For details, see section [2.2.1.7.4](#).

2.2.1.2.6 MONITOR_CONTAINER

The **MONITOR_CONTAINER** structure provides information about **port monitors**, using [MONITOR_INFO](#) structures. The **MonitorInfo** member specifies the structure that defines the port monitor properties.

```
typedef struct _MONITOR_CONTAINER {
    DWORD Level;
    [switch_is(Level)] union {
        [case(1)]
            MONITOR_INFO_1* pMonitorInfo1;
        [case(2)]
            MONITOR_INFO_2* pMonitorInfo2;
    } MonitorInfo;
} MONITOR_CONTAINER;
```

Level: This member MUST specify the information level of the **MonitorInfo** data. The value of this member MUST be 0x00000001 or 0x00000002.

MonitorInfo: This member MUST define port monitor properties, using an information structure that MUST correspond to the value of the **Level** member.

pMonitorInfo1: If the **Level** member is 1, this member MUST be a non-null pointer to a [MONITOR_INFO_1](#) structure that provides information about a port monitor. For details, see section [2.2.1.8.1](#).

pMonitorInfo2: If the **Level** member is 2, this member MUST be a non-null pointer to a [MONITOR_INFO_2](#) structure that provides information about a port monitor. For details, see section [2.2.1.8.2](#).

2.2.1.2.7 PORT_CONTAINER

The **PORT_CONTAINER** structure provides information about printer ports, using [PORT_INFO](#) structures. The **PortInfo** member specifies the structure that defines the port properties.

```
typedef struct _PORT_CONTAINER {
    DWORD Level;
    [switch_is(0x00FFFFFF & Level)]
    union {
        [case(1)]
            PORT_INFO_1* pPortInfo1;
        [case(2)]
```

```

    PORT_INFO_2* pPortInfo2;
[case(3)]
    PORT_INFO_3* pPortInfo3;
[case(0x00FFFFFF)]
    PORT_INFO_FF* pPortInfoFF;
} PortInfo;
} PORT_CONTAINER;

```

Level: This member MUST specify the information level of the **PortInfo** data. The value of this member MUST be in the range 0x00000001 to 0x00000003 inclusive, or 0xFFFFFFFF.

PortInfo: This member MUST define port properties, using an information structure that MUST correspond to the value of the **Level** member.

Note The bitwise AND of **Level** with 0x00FFFFFF does not indicate that any values for **Level** are valid besides those specified.

pPortInfo1: If the **Level** member is 1, this member MUST be a non-null pointer to a [PORT_INFO_1](#) structure that provides information about the printer port. For details, see section [2.2.1.9.1](#).

pPortInfo2: If the **Level** member is 2, this member MUST be a non-null pointer to a [PORT_INFO_2](#) structure that provides information about the printer port. For details, see section [2.2.1.9.2](#).

pPortInfo3: If the **Level** member is 3, this member MUST be a non-null pointer to a [PORT_INFO_3](#) structure that provides information about the printer port. For details, see section [2.2.1.9.3](#).

pPortInfoFF: If the **Level** member is 0xFFFFFFFF, this member MUST be a non-null pointer to a [PORT_INFO_FF](#) structure that provides information about the printer port. For details, see section [2.2.1.9.4](#).

2.2.1.2.8 PORT_VAR_CONTAINER

The **PORT_VAR_CONTAINER** structure provides information for supported printer port monitors.

```

typedef struct _PORT_VAR_CONTAINER {
    DWORD cbMonitorData;
    [size_is(cbMonitorData), unique, disable_consistency_check]
    BYTE* pMonitorData;
} PORT_VAR_CONTAINER;

```

cbMonitorData: This member MUST specify the size, in bytes, of the buffer that is pointed to by the **pMonitorData** member.

pMonitorData: This member MUST be null or it MUST be a non-null pointer to a block of data that is passed to the port monitor. This member MUST have the IDL attribute **disable_consistency_check**, so the byte buffer is not checked for consistency by RPC.

This attribute SHOULD be used for situations where the buffer content is unknown to the spooler process and may need to be passed on to a third-party object, such as a port monitor

or print processor. The consistency of the data MUST NOT be validated by the underlying **RPC engine**. For further information, see [\[MS-RPCE\]](#) section 3.1.1.5

2.2.1.2.9 PRINTER_CONTAINER

The **PRINTER_CONTAINER** structure provides information about printer properties and state information, using [PRINTER_INFO](#) structures. The **PrinterInfo** member specifies the structure that defines the printer properties.

```
typedef struct _PRINTER_CONTAINER {
    DWORD Level;
    [switch_is(Level)] union {
        [case(0)]
            PRINTER_INFO_STRESS* pPrinterInfoStress;
        [case(1)]
            PRINTER_INFO_1* pPrinterInfo1;
        [case(2)]
            PRINTER_INFO_2* pPrinterInfo2;
        [case(3)]
            PRINTER_INFO_3* pPrinterInfo3;
        [case(4)]
            PRINTER_INFO_4* pPrinterInfo4;
        [case(5)]
            PRINTER_INFO_5* pPrinterInfo5;
        [case(6)]
            PRINTER_INFO_6* pPrinterInfo6;
        [case(7)]
            PRINTER_INFO_7* pPrinterInfo7;
        [case(8)]
            PRINTER_INFO_8* pPrinterInfo8;
        [case(9)]
            PRINTER_INFO_9* pPrinterInfo9;
    } PrinterInfo;
} PRINTER_CONTAINER;
```

Level: This member MUST specify the information level of the **PrinterInfo** data. The value of this member MUST be in the range 0x00000000 to 0x00000009 inclusive.

PrinterInfo: This member provides the printer information using a container structure that MUST correspond to the value specified by the **Level** member.

pPrinterInfoStress: If the **Level** member is 0, this member MUST be a non-null pointer to a **PRINTER_INFO_STRESS** structure, which provides diagnostic printer information. For details, see section [2.2.1.10.1](#).

pPrinterInfo1: If the **Level** member is 1, this member MUST be a non-null pointer to a [PRINTER_INFO_1](#) structure, which provides printer information. For details, see section [2.2.1.10.2](#).

pPrinterInfo2: If the **Level** member is 2, this member MUST be a non-null pointer to a [PRINTER_INFO_2](#) structure, which provides detailed printer information. For details, see section [2.2.1.10.3](#).

pPrinterInfo3: If the **Level** member is 3, this member MUST be a non-null pointer to a [PRINTER_INFO_3](#) structure, which provides printer security information. For details, see section [2.2.1.10.4](#).

pPrinterInfo4: If the **Level** member is 4, this member MUST be a non-null pointer to a [PRINTER_INFO_4](#) structure, which provides a subset of the printer information. For details, see section [2.2.1.10.5](#).

pPrinterInfo5: If the **Level** member is 5, this member MUST be a non-null pointer to a [PRINTER_INFO_5](#) structure, which provides information about the printer attributes. For details, see section [2.2.1.10.6](#).

pPrinterInfo6: If the **Level** member is 6, this member MUST be a non-null pointer to a [PRINTER_INFO_6](#) structure, which provides information about the status of the printer. For details, see section [2.2.1.10.7](#).

pPrinterInfo7: If the **Level** member is 7, this member MUST be a non-null pointer to a [PRINTER_INFO_7](#) structure, which provides **directory services** information. For details, see section [2.2.1.10.8](#).

pPrinterInfo8: If the **Level** member is 8, this member MUST be a non-null pointer to a [PRINTER_INFO_8](#) structure, which provides information about the global printer driver settings for a printer. For details, see section [2.2.1.10.9](#).

pPrinterInfo9: If the **Level** member is 9, this member MUST be a non-null pointer to a [PRINTER_INFO_9](#) structure, which provides information about a per-user printer driver setting for a printer. For details, see section [2.2.1.10.10](#).

2.2.1.2.10 RPC_BIDI_REQUEST_CONTAINER

The **RPC_BIDI_REQUEST_CONTAINER** structure is a container for a list of **bidirectional** requests.

```
typedef struct _RPC_BIDI_REQUEST_CONTAINER {
    DWORD Version;
    DWORD Flags;
    DWORD Count;
    [size_is(Count), unique] RPC_BIDI_REQUEST_DATA aData[];
} RPC_BIDI_REQUEST_CONTAINER;
```

Version: This member MUST contain a value that specifies the version of the bidirectional API schema. The value of this member MUST be 0x00000001.

Flags: This member is reserved. The value of this member MUST be set to zero and it MUST be ignored upon receipt.

Count: This member MUST specify the number of bidirectional requests in the **aData** member.

aData: This member is an array of **RPC_BIDI_REQUEST_DATA** structures. Each structure in this member MUST contain a single bidirectional request. For details, see section [2.2.1.12.1](#).

2.2.1.2.11 RPC_BIDI_RESPONSE_CONTAINER

The **RPC_BIDI_RESPONSE_CONTAINER** structure is a container for a list of bidirectional responses.

```
typedef struct _RPC_BIDI_RESPONSE_CONTAINER {
    DWORD Version;
    DWORD Flags;
    DWORD Count;
    [size_is(Count), unique] RPC_BIDI_RESPONSE_DATA aData[];
} RPC_BIDI_RESPONSE_CONTAINER;
```

Version: This member MUST contain the value that specifies the version of the bidirectional API schema. The value of this member MUST be 0x00000001.

Flags: This member is a set of flags that are reserved for system use. The value of this member MUST be set to zero and it MUST be ignored upon receipt.

Count: This member MUST specify the number of bidirectional responses in the **aData** member.

aData: This member is an array of **RPC_BIDI_RESPONSE_DATA** structures. Each structure in this member MUST contain a single bidirectional response. For more information, see section [2.2.1.12.2](#).

2.2.1.2.12 RPC_BINARY_CONTAINER

The **RPC_BINARY_CONTAINER** structure is a container for binary printer data and is typically used in bidirectional responses.

```
typedef struct _RPC_BINARY_CONTAINER {
    DWORD cbBuf;
    [size_is(cbBuf), unique] BYTE* pszString;
} RPC_BINARY_CONTAINER;
```

cbBuf: This member MUST specify the size, in bytes, of the buffer that is pointed to by the **pszString** member.

pszString: This member MUST be a non-null pointer to an array of bytes that contain binary printer data.

2.2.1.2.13 SECURITY_CONTAINER

The **SECURITY_CONTAINER** structure is a container for [SECURITY_DESCRIPTOR](#) information.

```
typedef struct SECURITY_CONTAINER {
    DWORD cbBuf;
    [size_is(cbBuf), unique] BYTE* pSecurity;
} SECURITY_CONTAINER;
```

cbBuf: This member MUST contain the size, in bytes, of the buffer that is pointed to by the **pSecurity** member.

pSecurity: This member MUST be null or it MUST be a non-null pointer to a **SECURITY_DESCRIPTOR** structure that provides the security information. The **SECURITY_DESCRIPTOR** MUST be in self-relative form. The **SECURITY_DESCRIPTOR** data type is specified in [\[MS-DTYP\]](#).

2.2.1.2.14 SPLCLIENT_CONTAINER

The **SPLCLIENT_CONTAINER** structure contains a union that provides information about the connecting client.

```
typedef struct _SPLCLIENT_CONTAINER {
    DWORD Level;
    [switch_is(Level)] union {
        [case(1)]
            SPLCLIENT_INFO_1* pClientInfo1;
        [case(2)]
            unsigned __int64 pClientInfo2;
        [case(3)]
            SPLCLIENT_INFO_3* pClientInfo3;
    } ClientInfo;
} SPLCLIENT_CONTAINER;
```

Level: This member MUST specify the container structure that is used by the **ClientInfo** member for data. The value MUST be either 0x00000001 or 0x00000003.

ClientInfo: This member MUST contain client information using a container structure that MUST correspond to the value specified by the **Level** member.

pClientInfo1: If the **Level** member is 1, this member MUST be a non-null pointer to a [SPLCLIENT_INFO_1](#) structure that provides client information. For details, see section [2.2.1.11.1](#).

pClientInfo2: A 64-bit value that is reserved and MUST NOT be used.

pClientInfo3: If the **Level** member is 3, this member MUST be a non-null pointer to a [SPLCLIENT_INFO_3](#) structure that provides client information. For details, see section [2.2.1.11.2](#).

2.2.1.2.15 STRING_CONTAINER

The **STRING_CONTAINER** structure contains a string.

```
typedef struct _STRING_CONTAINER {
    DWORD cbBuf;
    [size_is(cbBuf/2), unique] WCHAR* pszString;
} STRING_CONTAINER;
```

cbBuf: This member MUST specify the size, in bytes, of the buffer that is pointed to by the **pszString** member. The value of this number MUST be an even number.

pszString: This member MUST be a non-null pointer to a string. The string that is referenced by this member MUST NOT be empty.

2.2.1.2.16 SYSTEMTIME_CONTAINER

The **SYSTEMTIME_CONTAINER** structure is a container for a [SYSTEMTIME](#) structure that specifies a date and time using individual members for the month, day, year, weekday, hour, minute, second, and millisecond.

```
typedef struct _SYSTEMTIME_CONTAINER {  
    DWORD cbBuf;  
    SYSTEMTIME* pSystemTime;  
} SYSTEMTIME_CONTAINER;
```

cbBuf: This member MUST specify the size, in bytes, of the buffer that is pointed to by the **pSystemTime** member.

pSystemTime: This member MUST be a non-null pointer to a **SYSTEMTIME** structure.

2.2.1.3 Members in INFO Structures

This section specifies members that are commonly used in a consistent fashion in IDL-marshaled and custom-marshaled [INFO](#) structures. The individual INFO sections provide definitions only for the following:

- Members that are not defined in this section;
- Members that are not defined in corresponding INFO subsections within this section; and
- Members whose definitions in their corresponding INFO structures differ from their definitions in this section and subsections.

The type of each member is specified in its corresponding INFO structure section.

pPrinterName: This member MUST be a non-null pointer to a [string](#) that MUST specify the name of a printer. For rules governing printer names, see section [2.2.4.2](#).

pServerName: This member MUST be a non-null pointer to a string that MUST specify the name of the server that hosts the printer. For rules governing server names, see section [2.2.4.1](#).

Reserved: This member is reserved for future use. The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

dwReserved2: This member is reserved for future use. The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

dwReserved3: This member is reserved for future use. The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

2.2.1.3.1 DRIVER_INFO and RPC_DRIVER_INFO Members

This section describes members commonly used in [DRIVER_INFO](#) and [RPC_DRIVER_INFO](#) structures.

pName: This member MUST be a non-null pointer to a string that MUST specify the name of the printer driver (for example, "QMS 810"). For rules governing printer driver names, see section [2.2.4.6](#).

cVersion: This member MUST contain an implementation-specific value that identifies the operating system (OS) version for which the printer driver was written. [<6>](#)

pEnvironment: This member MUST be a non-null pointer to a string that MUST specify the environment that the printer driver supports. For rules governing environment names and Windows behaviors, see section [2.2.4.5](#).

pDriverPath: This member MUST be a non-null pointer to a string that MUST specify a file name or full path and file name for the file that contains the printer driver. For further information on driver files, see [\[MSDN-MPD\]](#). For rules governing path names, see section [2.2.4.7.<7>](#)

pDataFile: This member MUST be a non-null pointer to a string that MUST specify a file name or a full path and file name for the file that contains printer driver data. For further information on driver files, see [\[MSDN-MPD\]](#). For rules governing path names, see section [2.2.4.7.<8>](#)

pConfigFile: This member MUST be a non-null pointer to a string that MUST specify a file name or a full path and file name for the printer driver configuration module. For further information on driver files, see [\[MSDN-MPD\]](#). For rules governing path names, see section [2.2.4.7.<9>](#)

pHelpFile: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify a file name or a full path and file name for the printer driver help file. For further information on driver files, see [\[MSDN-MPD\]](#). For rules governing path names, see section [2.2.4.7.<10>](#)

pMonitorName: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify a **language monitor**. For rules governing monitor names, see section [2.2.4.12.<11>](#)

pDefaultDataType: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify the default datatype of print jobs created with this driver (for example, "**EMF**" or "**RAW**"). For rules governing datatype names, see section [2.2.4.15](#).

cchDependentFiles: The value of this member MUST specify the number of characters in the multisz pointed to by **pDependentFiles**.

pDependentFiles: This member MUST be null or it MUST be a non-null pointer to a multisz that MUST specify the files that the printer driver is dependent on. This list MUST include one or more files. [<12>](#)

cchPreviousNames: The value of this member MUST be the number of characters in the multisz pointed to by **pszzPreviousNames**.

pszzPreviousNames: This member MUST be null or it MUST be a non-null pointer to a multisz that MUST specify any previous printer drivers that are compatible with this driver. [<13>](#)

dwDriverVersion: The value of this member MUST specify the printer driver version number. The format of this number is specified by each printer driver manufacturer. [<14>](#)

ftDriverDate: The value of this member MUST be the manufacturer build date of the printer driver. The FILETIME format is specified in [\[MS-DTYP\]](#) section **2.1.6**.

pMfgName: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify the manufacturer's name.

pOEMUrl: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify the URL for the printer driver's manufacturer.

pHardwareID: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify the hardware identifier for the printer driver. [<15>](#)

pProvider: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify the publisher of the printer driver. [<16>](#)

2.2.1.3.2 FORM_INFO and RPC_FORM_INFO Members

This section describes the members that are commonly used in [FORM_INFO](#) and [RPC_FORM_INFO](#) structures.

Flags: This member MUST specify the form property. The value of this member MUST be a value from the following table.

Name/Value	Meaning
FORM_USER 0x00000000	If the value of this member is FORM_USER , the form has been defined by the user. Forms that have this flag set are defined in the registry.
FORM_BUILTIN 0x00000001	If the value of this member is FORM_BUILTIN , the form is part of the spooler. Form definitions that have this flag set do not appear in the registry.
FORM_PRINTER 0x00000002	If the value of this member is FORM_PRINTER , the form is associated with a particular printer and its definition appears in the registry.

pName: This member MUST be a non-null pointer to a string that MUST specify the form name. For rules governing form names, see section [2.2.4.11](#).

Size: The value of this member MUST specify the form's width and height in thousandths of millimeters using a [SIZE](#) structure.

ImageableArea: This member MUST specify the part of the form that the printer can print on as a rectangle in thousandths of millimeters using a [RECTL](#) structure.

2.2.1.3.3 JOB_INFO Members

This section describes members commonly used in [JOB_INFO](#) structures.

pMachineName: This member MUST be a non-null pointer to a string that MUST specify the name of the server that hosts the printer. For rules governing server names, see section [2.2.4.1](#).

pUserName: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify the name of the user that owns the print job. For rules governing user names, see section [2.2.4.18](#).

pNotifyName: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify the name of the user to be notified when the job is complete or when an error occurs while printing the job. For rules governing user names, see section [2.2.4.18](#).

pDocument: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify the name of the print job.

pDatatype: This member MUST be a non-null pointer to a string that MUST specify the type of data that the printing application sends in the print job to the printer. The identified data type MUST be valid and supported by the print processor that is associated with the printer that is processing the job. For rules governing datatype names, see section [2.2.4.15](#).

pPrintProcessor: This member MUST be a non-null pointer to a string that MUST specify the name of the print processor that MUST be used to print the job. For rules governing print processor names, see section [2.2.4.8](#).

pParameters: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify the default print processor parameters.

pDriverName: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify the name of the printer driver that MUST be used to process the print job. For rules governing printer driver names, see section [2.2.4.6](#).

pDevMode: This member MUST be null or it MUST be a non-null pointer to a **DEVMODE** structure defining default printer data. For more information about the **DEVMODE** structure, see section [2.2.2.1](#) and [\[DEVMODE\]](#).

pSecurityDescriptor: The value of this member SHOULD be null and MUST be ignored upon receipt.

JobId: This member MUST contain a non-null identifier for the print job.

pStatus: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify a textual representation of the job status. The textual representation is implementation-specific and MAY be displayed to the user but MUST NOT have any other functional effect. An example of a textual representation is "Cannot print - Black ink needs to be replaced."

Status: This member specifies the job status. The value of this member MUST be the result of a bitwise OR of zero or more of the job status values defined in section [2.2.3.1](#).

Client applications MAY display a textual representation of the job status to the user. The textual representation is an implementation-specific string and SHOULD support all job status descriptions specified in section [2.2.3.1](#) for all corresponding status bits. If **pStatus** is not null, the string that is pointed to by **pStatus** SHOULD be displayed instead.

Priority: This member MUST specify information about the job priority. The value of this member MUST be a decimal number from 0 through 99, inclusive.

Position: This member MUST specify the job's position in the queue where one represents the next job that will be printed.

TotalPages: The value of this member MUST specify the number of pages the document contains. The value of this member MAY be zero.

PagesPrinted: The value of this member MUST specify the number of pages that have been printed. The value of this member MAY be zero.

Submitted: This member is a [SYSTEMTIME](#) structure that MUST specify when the document was spooled.

StartTime: This member MUST specify the earliest time that the printer can print a job. The time is expressed as the number of minutes after 12:00 AM GMT within a 24-hour boundary.

UntilTime: This member MUST specify the latest time that the printer can print a job. The time is expressed as the number of minutes after 12:00 AM GMT within a 24-hour boundary.

Size: The value of this member MUST specify the size of the job in bytes.

Time: The value of this member MUST specify the number of milliseconds that have elapsed since printing began.

2.2.1.3.4 MONITOR_INFO Members

This section describes the members that are commonly used in [MONITOR_INFO](#) structures.

pName: This member MUST be a non-null pointer to a string that MUST specify the name of the port monitor. For rules governing port monitor names, see section [2.2.4.12](#).

2.2.1.3.5 PORT_INFO Members

This section describes members commonly used in [PORT_INFO](#) structures.

pPortName: This member MUST be a non-null pointer to a string that MUST specify a supported printer port. For rules governing port names, see section [2.2.4.13](#).

2.2.1.3.6 PRINTER_INFO Members

This section describes members commonly used in [PRINTER_INFO](#) structures.

pDescription: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify a description of the printer. [<17>](#)

pComment: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify additional information about the printer. [<18>](#)

Status: The value of this member MUST specify the printer status. It MUST be the result of a bitwise OR of zero or more of the printer status values defined in section [2.2.3.1](#).

Attributes: The value of this member MUST specify printer attributes. It MUST be the result of a bitwise OR of zero or more of the printer attribute values defined in section [2.2.3.1](#).

pSecurityDescriptor: This member MUST be null, or it MUST be a non-null pointer to a [SECURITY_DESCRIPTOR](#) structure, as specified in [\[MS-DTYP\]](#) in self-relative memory format that MUST specify a printer's security information.

pPortName: This member MUST be a non-null pointer to a string that MUST specify the port(s) used to transmit data to the printer. For rules governing port names, see section [2.2.4.13](#).

2.2.1.3.7 SPLCLIENT_INFO Members

This section describes members commonly used in [SPLCLIENT_INFO](#) structures.

pMachineName: This member MUST be a non-null pointer to a string that MUST provide the client computer name. Client computer names are governed by the same rules as server names. For rules governing server names, see section [2.2.4.1](#).

pUserName: This member MUST be a non-null pointer to a string that MUST provide a user name.

dwBuildNum: The value of this member MUST specify the build number of the client operating system.

dwMajorVersion: The value of this member MUST specify the major version number of the client operating system. [<19>](#)

dwMinorVersion: The value of this member MUST specify the minor version number of the client operating system. [<20>](#)

wProcessorArchitecture: The value of this member MUST specify the implementation-specific identifier for the client system's processor architecture. The value of this member SHOULD be ignored upon receipt. [<21>](#)

2.2.1.4 DOC_INFO_1

The **DOC_INFO_1** structure describes a document that will be printed.

```
typedef struct _DOC_INFO_1 {  
    [string] wchar_t* pDocName;  
    [string] wchar_t* pOutputFile;  
    [string] wchar_t* pDataType;  
} DOC_INFO_1;
```

pDocName: This member MAY be null or MUST be a non-null pointer to a string that MUST provide the name of the document. If this member is null, the print server SHOULD use an implementation-specific default job name.

pOutputFile: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify the name of an output file. For rules governing path names, see section [2.2.4.7](#).

pDataType: This member MUST be null or it MUST be a non-null pointer to a string that MUST identify the type of data used to record the document. For rules governing datatype names, see section [2.2.4.15](#).

2.2.1.5 DRIVER_INFO

2.2.1.5.1 DRIVER_INFO_1

The **DRIVER_INFO_1** structure provides information about a printer driver.

```
typedef struct _DRIVER_INFO_1 {  
    [string] wchar_t* pName;  
} DRIVER_INFO_1;
```

All members not defined in this section are specified in sections [2.2.1.3.1](#) and [2.2.1.3](#).

2.2.1.5.2 DRIVER_INFO_2

The **DRIVER_INFO_2** structure provides information about a printer driver.

```
typedef struct _DRIVER_INFO_2 {  
    DWORD cVersion;  
    [string] wchar_t* pName;  
    [string] wchar_t* pEnvironment;
```

```

    [string] wchar_t* pDriverPath;
    [string] wchar_t* pDataFile;
    [string] wchar_t* pConfigFile;
} DRIVER_INFO_2;

```

All members not defined in this section are specified in sections [2.2.1.3.1](#) and [2.2.1.3](#).

2.2.1.5.3 RPC_DRIVER_INFO_3

The **RPC_DRIVER_INFO_3** structure provides information about a printer driver.

```

typedef struct _RPC_DRIVER_INFO_3 {
    DWORD cVersion;
    [string] wchar_t* pName;
    [string] wchar_t* pEnvironment;
    [string] wchar_t* pDriverPath;
    [string] wchar_t* pDataFile;
    [string] wchar_t* pConfigFile;
    [string] wchar_t* pHelpFile;
    [string] wchar_t* pMonitorName;
    [string] wchar_t* pDefaultDataType;
    DWORD cchDependentFiles;
    [size_is(cchDependentFiles), unique]
    wchar_t* pDependentFiles;
} RPC_DRIVER_INFO_3;

```

All members not defined in this section are specified in sections [2.2.1.3.1](#) and [2.2.1.3](#).

2.2.1.5.4 RPC_DRIVER_INFO_4

The **RPC_DRIVER_INFO_4** structure provides information about a printer driver.

```

typedef struct _RPC_DRIVER_INFO_4 {
    DWORD cVersion;
    [string] wchar_t* pName;
    [string] wchar_t* pEnvironment;
    [string] wchar_t* pDriverPath;
    [string] wchar_t* pDataFile;
    [string] wchar_t* pConfigFile;
    [string] wchar_t* pHelpFile;
    [string] wchar_t* pMonitorName;
    [string] wchar_t* pDefaultDataType;
    DWORD cchDependentFiles;
    [size_is(cchDependentFiles), unique]
    wchar_t* pDependentFiles;
    DWORD cchPreviousNames;
    [size_is(cchPreviousNames), unique]
    wchar_t* pszzPreviousNames;
} RPC_DRIVER_INFO_4;

```

All members not defined in this section are specified in sections [2.2.1.3.1](#) and [2.2.1.3](#).

2.2.1.5.5 RPC_DRIVER_INFO_6

The **RPC_DRIVER_INFO_6** structure provides extended printer driver information.

```
typedef struct _RPC_DRIVER_INFO_6 {
    DWORD cVersion;
    [string] wchar_t* pName;
    [string] wchar_t* pEnvironment;
    [string] wchar_t* pDriverPath;
    [string] wchar_t* pDataFile;
    [string] wchar_t* pConfigFile;
    [string] wchar_t* pHelpFile;
    [string] wchar_t* pMonitorName;
    [string] wchar_t* pDefaultDataType;
    DWORD cchDependentFiles;
    [size_is(cchDependentFiles), unique]
        wchar_t* pDependentFiles;
    DWORD cchPreviousNames;
    [size_is(cchPreviousNames), unique]
        wchar_t* pszzPreviousNames;
    FILETIME ftDriverDate;
    DWORDLONG dwlDriverVersion;
    [string] wchar_t* pMfgName;
    [string] wchar_t* pOEMUrl;
    [string] wchar_t* pHardwareID;
    [string] wchar_t* pProvider;
} RPC_DRIVER_INFO_6;
```

All members not defined in this section are specified in sections [2.2.1.3.1](#) and [2.2.1.3](#).

2.2.1.5.6 RPC_DRIVER_INFO_8

The **RPC_DRIVER_INFO_8** structure specifies extended printer driver information.

```
typedef struct _RPC_DRIVER_INFO_8 {
    DWORD cVersion;
    [string] wchar_t* pName;
    [string] wchar_t* pEnvironment;
    [string] wchar_t* pDriverPath;
    [string] wchar_t* pDataFile;
    [string] wchar_t* pConfigFile;
    [string] wchar_t* pHelpFile;
    [string] wchar_t* pMonitorName;
    [string] wchar_t* pDefaultDataType;
    DWORD cchDependentFiles;
    [size_is(cchDependentFiles), unique]
        wchar_t* pDependentFiles;
    DWORD cchPreviousNames;
    [size_is(cchPreviousNames), unique]
        wchar_t* pszzPreviousNames;
    FILETIME ftDriverDate;
    DWORDLONG dwlDriverVersion;
    [string] wchar_t* pMfgName;
```

```

[string] wchar_t* pOEMUrl;
[string] wchar_t* pHardwareID;
[string] wchar_t* pProvider;
[string] wchar_t* pPrintProcessor;
[string] wchar_t* pVendorSetup;
DWORD cchColorProfiles;
[size_is(cchColorProfiles), unique]
    wchar_t* pszzColorProfiles;
[string] wchar_t* pInfPath;
DWORD dwPrinterDriverAttributes;
DWORD cchCoreDependencies;
[size_is(cchCoreDependencies), unique]
    wchar_t* pszzCoreDriverDependencies;
FILETIME ftMinInboxDriverVerDate;
DWORDLONG dwlMinInboxDriverVerVersion;
} RPC_DRIVER_INFO_8;

```

pPrintProcessor: This member MUST be a non-null pointer to a string that MUST specify the print processor for this printer. For rules governing print processor names, see section [2.2.4.8](#).

pVendorSetup: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify the name of the vendor setup file used for hardware vendor-provided custom setup.

cchColorProfiles: The value of this member MUST specify the number of characters that are in the multisz that is pointed to by the **pszzColorProfiles** member.

pszzColorProfiles: This member MUST be null or it MUST be a non-null pointer to a multisz that MUST contain the names of all **color profile** files for this driver.

pInfPath: This member MUST be a non-null pointer to a **string** that identifies the path to the installation configuration file in the **driver store** that MUST identify the printer driver for installation. [<22>](#)

dwPrinterDriverAttributes: The value of this member MUST specify the attributes of the printer driver. The value of this member MUST either be set to zero or contain the following value:

Name/Value	Meaning
PRINTER_DRIVER_PACKAGE_AWARE 0x00000001	The printer driver is part of a driver package.

cchCoreDependencies: The value of this member MUST specify the number of characters that are in the multisz that is pointed to by the **pszzCoreDriverDependencies** member.

pszzCoreDriverDependencies: This member MUST be null or it MUST be a non-null pointer to a multisz that MUST contain the names of the core dependencies as specified by the installation configuration file. These names MUST specify the core sections of the installation configuration file that are required by the printer driver. [<23>](#)

ftMinInboxDriverVerDate: The value of this member applies to only package-aware printer drivers. The value of this member MUST specify the minimum date version that is required for any in-box, **core printer driver** dependencies that are listed in the multisz that is pointed to

by the **pszzCoreDriverDependencies** member. The value of this member MUST be specified in the same format as the value of the **ftDriverDate** member.[<24>](#)

dwlMinInboxDriverVerVersion: The value of this member MUST specify the minimum file version that is required for any in-box core printer driver dependencies that are listed in the multisz that is pointed to by the **pszzCoreDriverDependencies** member. The value of this member MUST be specified in the same format as the value of the **dwlDriverVersion** member.[<25>](#)

All members not defined in this section are specified in sections [2.2.1.3.1](#) and [2.2.1.3](#).

2.2.1.6 FORM_INFO

2.2.1.6.1 FORM_INFO_1

The **FORM_INFO_1** structure provides information about a printer form.

```
typedef struct _FORM_INFO_1 {
    DWORD Flags;
    [string] wchar_t* pName;
    SIZE Size;
    RECTL ImageableArea;
} FORM_INFO_1;
```

All members not defined in this section are specified in sections [2.2.1.3.1](#) and [2.2.1.3](#).

2.2.1.6.2 RPC_FORM_INFO_2

The **RPC_FORM_INFO_2** structure provides information about a printer form that includes its origin, dimensions, the dimensions of its printable area, and its display name.

```
typedef struct _RPC_FORM_INFO_2 {
    DWORD Flags;
    [string, unique] const wchar_t* pName;
    SIZE Size;
    RECTL ImageableArea;
    [string, unique] const char* pKeyword;
    DWORD StringType;
    [string, unique] const wchar_t* pMuiDll;
    DWORD dwResourceId;
    [string, unique] const wchar_t* pDisplayName;
    LANGID wLangID;
} RPC_FORM_INFO_2;
```

pKeyword: This member MUST be set to null by the client if the value of the **Flags** member is set to **FORM_BUILTIN**, or this member MUST be a non-null pointer to a string that MUST specify a unique, localization-independent identifier for this form.[<26>](#)

StringType: The value of this member MUST specify how a form's display name is passed. The value of this member MUST be a value from the following table.

Name/Value	Meaning
STRING_NONE 0x00000001	Use the default display name, a string that is pointed to by the pName member. No localized display name exists.
STRING_MUIDLL 0x00000002	Load the form name from the DLL that is identified by the string that is pointed to by the pMuidll member. The value of the dwResourceId member specifies the resource ID of the form name that is located in the DLL that is identified by the string that is pointed to by the pMuidll member.
STRING_LANGPAIR 0x00000004	Use the form name, a string that is pointed to by the pDisplayName member, and the language that is identified by the wLangID member.

pMuidll: This member MUST be a null pointer and MUST be ignored upon receipt if **StringType** is not equal to STRING_MUIDLL, or it MUST be a non-null pointer to a string that contains the name of a localized resources DLL.

dwResourceId: The value of this member SHOULD be zero and ignored upon receipt if the value of the **StringType** member is not equal to STRING_MUIDLL; otherwise, the value of this member MUST specify the resource ID of the form name in the DLL that is identified by the string that is pointed to by the **pMuidll** member.

pDisplayName: This member MUST be a null pointer and ignored upon receipt if **StringType** is not equal to STRING_LANGPAIR; otherwise, this member MUST be a non-null pointer to a string that MUST specify the form name.

wLangID: The value of this member SHOULD be zero and ignored upon receipt if **StringType** is not equal to STRING_LANGPAIR; otherwise, the value of this member MUST be the Language Identifier of the **pDisplayName** member as specified in [\[MS-LCID\]](#).

All members not defined in this section are specified in sections [2.2.1.3.2](#) and [2.2.1.3](#).

2.2.1.7 JOB_INFO

2.2.1.7.1 JOB_INFO_1

The **JOB_INFO_1** structure provides information about a print job.

```
typedef struct _JOB_INFO_1 {
    DWORD JobId;
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pMachineName;
    [string] wchar_t* pUserName;
    [string] wchar_t* pDocument;
    [string] wchar_t* pDataatype;
    [string] wchar_t* pStatus;
    DWORD Status;
    DWORD Priority;
    DWORD Position;
    DWORD TotalPages;
    DWORD PagesPrinted;
    SYSTEMTIME Submitted;
} JOB_INFO_1;
```

All members not defined in this section are specified in sections [2.2.1.3.3](#) and [2.2.1.3](#).

2.2.1.7.2 JOB_INFO_2

The **JOB_INFO_2** structure provides information about a print job.

```
typedef struct _JOB_INFO_2 {
    DWORD JobId;
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pMachineName;
    [string] wchar_t* pUserName;
    [string] wchar_t* pDocument;
    [string] wchar_t* pNotifyName;
    [string] wchar_t* pDataatype;
    [string] wchar_t* pPrintProcessor;
    [string] wchar_t* pParameters;
    [string] wchar_t* pDriverName;
    DEVMODE* pDevMode;
    [string] wchar_t* pStatus;
    SECURITY_DESCRIPTOR* pSecurityDescriptor;
    DWORD Status;
    DWORD Priority;
    DWORD Position;
    DWORD StartTime;
    DWORD UntilTime;
    DWORD TotalPages;
    DWORD Size;
    SYSTEMTIME Submitted;
    DWORD Time;
    DWORD PagesPrinted;
} JOB_INFO_2;
```

All members not defined in this section are specified in sections [2.2.1.3.3](#) and [2.2.1.3](#).

2.2.1.7.3 JOB_INFO_3

The **JOB_INFO_3** structure provides information about a print job.

```
typedef struct _JOB_INFO_3 {
    DWORD JobId;
    DWORD NextJobId;
    DWORD Reserved;
} JOB_INFO_3;
```

NextJobId: The value of this member MUST be zero, or it MUST be an identifier that specifies the print job in the queue that follows the print job that is identified by the **JobId** member. A value of zero indicates that there are no jobs that follow the job identified by the **JobId** member.

When used as input to [RpcSetJob](#) to alter the order of print jobs, **JobId** and **NextJobId** MAY be obtained through [RpcEnumJobs](#) or [RpcGetJob](#).

All members not defined in this section are specified in sections [2.2.1.3.3](#) and [2.2.1.3](#).

2.2.1.7.4 JOB_INFO_4

The **JOB_INFO_4** structure provides information about a print job.

```
typedef struct _JOB_INFO_4 {
    DWORD JobId;
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pMachineName;
    [string] wchar_t* pUserName;
    [string] wchar_t* pDocument;
    [string] wchar_t* pNotifyName;
    [string] wchar_t* pDataatype;
    [string] wchar_t* pPrintProcessor;
    [string] wchar_t* pParameters;
    [string] wchar_t* pDriverName;
    DEVMODE* pDevMode;
    [string] wchar_t* pStatus;
    SECURITY_DESCRIPTOR* pSecurityDescriptor;
    DWORD Status;
    DWORD Priority;
    DWORD Position;
    DWORD StartTime;
    DWORD UntilTime;
    DWORD TotalPages;
    DWORD Size;
    SYSTEMTIME Submitted;
    DWORD Time;
    DWORD PagesPrinted;
    long SizeHigh;
} JOB_INFO_4;
```

SizeHigh: The value of this member **MUST** specify the high-order 32 bits of the 64-bit unsigned integer that indicates the size of the job in bytes.

All members not defined in this section are specified in sections [2.2.1.3.3](#) and [2.2.1.3](#).

2.2.1.8 MONITOR_INFO

2.2.1.8.1 MONITOR_INFO_1

The **MONITOR_INFO_1** structure provides information about a monitor.

```
typedef struct _MONITOR_INFO_1 {
    [string] wchar_t* pName;
} MONITOR_INFO_1;
```

All members not defined in this section are specified in sections [2.2.1.3.4](#) and [2.2.1.3](#).

2.2.1.8.2 MONITOR_INFO_2

The **MONITOR_INFO_2** structure provides information about a monitor.

```
typedef struct _MONITOR_INFO_2 {
    [string] wchar_t* pName;
    [string] wchar_t* pEnvironment;
    [string] wchar_t* pDLLName;
} MONITOR_INFO_2;
```

pEnvironment: This member MUST be a non-null pointer to a string that MUST specify the environment that the monitor supports. The environment specified MUST match the print server's operating system. For rules governing environment names and Windows behaviors, see section [2.2.4.5](#).

pDLLName: This member MUST be a non-null pointer to a string that MUST specify the name of the port monitor executable object.

All members not defined in this section are specified in sections [2.2.1.3.4](#) and [2.2.1.3](#).

2.2.1.9 PORT_INFO

2.2.1.9.1 PORT_INFO_1

The **PORT_INFO_1** structure provides information about a port.

```
typedef struct _PORT_INFO_1 {
    [string] wchar_t* pName;
} PORT_INFO_1;
```

pName: This member is synonymous with the **pPortName** member.

All members not defined in this section are specified in sections [2.2.1.3.5](#) and [2.2.1.3](#).

2.2.1.9.2 PORT_INFO_2

The **PORT_INFO_2** structure provides information about a port.

```
typedef struct _PORT_INFO_2 {
    [string] wchar_t* pPortName;
    [string] wchar_t* pMonitorName;
    [string] wchar_t* pDescription;
    DWORD fPortType;
    DWORD Reserved;
} PORT_INFO_2;
```

pMonitorName: This member MUST be a non-null pointer to a string that MUST specify an installed port monitor. For rules governing port monitor names, see section [2.2.4.12](#).

pDescription: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify implementation-specific additional information about the printer port. [<27>](#)

fPortType: The value of this member MUST be a bitmask that describes a printer port. The value of this member MUST be the result of a bitwise OR of zero or more of the following values:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	D	C	B	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Where the bits are defined as:

Value	Description
A PORT_TYPE_WRITE	The port can be written to.
B PORT_TYPE_READ	The port can be read from.
C PORT_TYPE_REDIRECTED	The port is a Terminal services redirected port.
D PORT_TYPE_NET_ATTACHED	The port is a network TCP/IP port.

All members not defined in this section are specified in sections [2.2.1.3.5](#) and [2.2.1.3](#).

2.2.1.9.3 PORT_INFO_3

The **PORT_INFO_3** structure provides information about a port.

```
typedef struct _PORT_INFO_3 {
    DWORD dwStatus;
    [string] wchar_t* pszStatus;
    DWORD dwSeverity;
} PORT_INFO_3;
```

dwStatus: The value of this member MUST specify the new port status. The value of this member MUST be one of the constant values in the following table.

Name/Value	Meaning
PORT_STATUS_CLEAR 0x00000000	Clears the printer port status.
PORT_STATUS_OFFLINE 0x00000001	The port's printer is offline.
PORT_STATUS_PAPER_JAM 0x00000002	The port's printer has a paper jam.

Name/Value	Meaning
PORT_STATUS_PAPER_OUT 0x00000003	The port's printer is out of paper.
PORT_STATUS_OUTPUT_BIN_FULL 0x00000004	The port's printer's output bin is full.
PORT_STATUS_PAPER_PROBLEM 0x00000005	The port's printer has a paper problem.
PORT_STATUS_NO_TONER 0x00000006	The port's printer is out of toner.
PORT_STATUS_DOOR_OPEN 0x00000007	The door of the port's printer is open.
PORT_STATUS_USER_INTERVENTION 0x00000008	The port's printer requires user intervention.
PORT_STATUS_OUT_OF_MEMORY 0x00000009	The port's printer is out of memory.
PORT_STATUS_TONER_LOW 0x0000000A	The port's printer is low on toner.
PORT_STATUS_WARMING_UP 0x0000000B	The port's printer is warming up.
PORT_STATUS_POWER_SAVE 0x0000000C	The port's printer is in a power-conservation mode.

pszStatus: This member MUST be null or it MUST be a non-null pointer to a string that MUST specify a status description.

dwSeverity: The value of this member MUST specify the port status value. The value of this member MUST be one of the constant values from the following table.

Name/Value	Meaning
PORT_STATUS_TYPE_ERROR 1	The port status value indicates an error.
PORT_STATUS_TYPE_WARNING 2	The port status value is a warning.
PORT_STATUS_TYPE_INFO 3	The port status value is informational.

All members not defined in this section are specified in sections [2.2.1.3.5](#) and [2.2.1.3](#).

2.2.1.9.4 PORT_INFO_FF

The **PORT_INFO_FF** is used to communicate port information to a local port monitor.

```
typedef struct _PORT_INFO_FF {
    wchar_t* pName;
```

```

    DWORD cbMonitorData;
    BYTE* pMonitorData;
} PORT_INFO_FF;

```

pName: This member is synonymous with the **pPortName** member.

cbMonitorData: This member is reserved for future use. The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

pMonitorData: This member is reserved for future use. This member SHOULD be a null pointer and MUST be ignored upon receipt.

All members not defined in this section are specified in sections [2.2.1.3.5](#) and [2.2.1.3](#).

2.2.1.10 PRINTER_INFO

2.2.1.10.1 PRINTER_INFO_STRESS

The **PRINTER_INFO_STRESS** structure provides diagnostic printer information used for **print system remote protocol stress analysis**.

```

typedef struct _PRINTER_INFO_STRESS {
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pServerName;
    DWORD cJobs;
    DWORD cTotalJobs;
    DWORD cTotalBytes;
    SYSTEMTIME stUpTime;
    DWORD MaxcRef;
    DWORD cTotalPagesPrinted;
    DWORD dwGetVersion;
    DWORD fFreeBuild;
    DWORD cSpooling;
    DWORD cMaxSpooling;
    DWORD cRef;
    DWORD cErrorOutOfPaper;
    DWORD cErrorNotReady;
    DWORD cJobError;
    DWORD dwNumberOfProcessors;
    DWORD dwProcessorType;
    DWORD dwHighPartTotalBytes;
    DWORD cChangeID;
    DWORD dwLastError;
    DWORD Status;
    DWORD cEnumerateNetworkPrinters;
    DWORD cAddNetPrinters;
    unsigned short wProcessorArchitecture;
    unsigned short wProcessorLevel;
    DWORD cRefIC;
    DWORD dwReserved2;
    DWORD dwReserved3;
} PRINTER_INFO_STRESS;

```

cJobs: The value of this member MUST specify the number of jobs that are currently in the print queue.

cTotalJobs: The value of this member MUST specify the total number of jobs that have been spooled since the print server was started.

cTotalBytes: The value of this member MUST contain the low-order 32 bits of the unsigned 64-bit value that specifies the total number of bytes that have been printed since system startup. The high-order 32 bits of the total number of bytes printed are supplied by the **dwHighPartTotalBytes** member.

stUpTime: This member MUST specify the time the printer data structure was created, in [SYSTEMTIME](#) format.

MaxcRef: The value of this member MUST specify the historic maximum value of the **cRef** member.

cTotalPagesPrinted: The value of this member MUST specify the total number of pages printed.

dwGetVersion: The value of this member MUST specify an implementation-specific value that represents the operating system version. [<28>](#)

fFreeBuild: The value of this member is an implementation-specific value that SHOULD be set to zero if the print server build contains extra debugging information and validation code and SHOULD be set to any nonzero value if the print server does not. The value of this member MUST be ignored upon receipt. [<29>](#)

cSpooling: The value of this member MUST specify the number of actively spooling jobs.

cMaxSpooling: The value of this member MUST specify the historic maximum number of actively spooling jobs.

cRef: The value of this member MUST specify the reference count for opened printer objects.

cErrorOutOfPaper: The value of this member MUST specify the total number of out-of-paper errors.

cErrorNotReady: The value of this member MUST specify the total number of not-ready errors.

cJobError: The value of this member MUST specify the total number of job errors.

dwNumberOfProcessors: The value of this member MUST specify the number of processors in the computer on which the print server is running.

dwProcessorType: The value of this member MUST specify the type of processor in the computer. [<30>](#)

dwHighPartTotalBytes: This member MUST contain the high-order 32 bits of the unsigned 64-bit value of the total number of bytes printed since system startup. The **cTotalBytes** member contains the low-order 32 bits of this value.

cChangeID: The value of this member MUST specify a unique number that identifies the last change.

dwLastError: The value of this member MUST specify an error code for the last error that occurred with this printer. [<31>](#)

Status: The value of this member MUST specify the current printer status. Details on printer status values are as specified in section [2.2.2.12](#).

cEnumerateNetworkPrinters: The value of this member MUST specify the number of times the network printers browse list has been requested.

cAddNetPrinters: The value of this member MUST specify the number of network printers added, per server.

wProcessorArchitecture: The value of this member MUST specify an implementation-specific value that identifies the system's processor architecture. This value SHOULD be ignored upon receipt.[<32>](#)

wProcessorLevel: The value of this member MUST specify an implementation-specific value that identifies the system's architecture-dependent processor level. This value SHOULD be ignored upon receipt.[<33>](#)

cRefIC: The value of this member MUST specify the number of open information context handles.

All members not defined in this section are specified in sections [2.2.1.3.6](#) and [2.2.1.3](#).

2.2.1.10.2 PRINTER_INFO_1

The **PRINTER_INFO_1** structure provides information about a printer.

```
typedef struct _PRINTER_INFO_1 {
    DWORD Flags;
    [string] wchar_t* pDescription;
    [string] wchar_t* pName;
    [string] wchar_t* pComment;
} PRINTER_INFO_1;
```

Flags: The value of this member MUST be the result of a bitwise OR of zero or more of the following values:

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	
X	X	X	X	X	X	X	X	C	E	X	X	X	X	X	X	I 8	X	X	X	X	I 3	I 2	I 1	X	X	X	X	X	X	X	X	H

Value	Description
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.

Value	Description
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
C	PRINTER_ENUM_CONTAINER : A single bit that, when set, MUST indicate that the printer object is capable of containing enumerable objects. For example, one such object is a print server that contains printers, or a print provider .
E	PRINTER_ENUM_EXPAND : A single bit that, when set, MUST indicate that the object contains further enumerable child objects. When RpcEnumPrinters is used to enumerate print servers, servers MUST set this bit for each enumerated server whose name matches a domain name.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
I8	PRINTER_ENUM_ICON8 : A single bit that, when set, MUST indicate that an application SHOULD treat the object as a print server. A GUI application MAY choose to display an icon of choice for this type of object.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
I3	A single bit that, when set, MUST indicate that (where appropriate) an application SHOULD treat the object as a print server. A GUI application MAY 34 choose to display an icon of choice for this type of object.
I2	PRINTER_ENUM_ICON2 : A single bit that, when set, MUST indicate that (where appropriate) an application SHOULD treat the object as a network domain name. A GUI application MAY 35 choose to display an icon of choice for this type of object.
I1	PRINTER_ENUM_ICON1 : A single bit that, when set, MUST indicate that (where appropriate) an application SHOULD treat the object as a top-level network name, such as Windows network. A GUI application MAY 36 choose to display an icon of choice for this type of object.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.

Value	Description
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
X	Reserved. The bit SHOULD be clear (zero) and MUST be ignored upon receipt.
H	PRINTER_ENUM_HIDE: A single bit that, when set, MUST indicate that an application SHOULD NOT display the object.

pName: This member is synonymous with [pPrinterName](#), as specified in section [3.1.4.1.1](#).

All members not defined in this section are specified in sections [2.2.1.3.6](#) and [2.2.1.3](#).

2.2.1.10.3 PRINTER_INFO_2

The **PRINTER_INFO_2** structure provides information about a printer.

```
typedef struct _PRINTER_INFO_2 {
    [string] wchar_t* pServerName;
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pShareName;
    [string] wchar_t* pPortName;
    [string] wchar_t* pDriverName;
    [string] wchar_t* pComment;
    [string] wchar_t* pLocation;
    DEVMODE* pDevMode;
    [string] wchar_t* pSepFile;
    [string] wchar_t* pPrintProcessor;
    [string] wchar_t* pDatatype;
    [string] wchar_t* pParameters;
    SECURITY_DESCRIPTOR* pSecurityDescriptor;
    DWORD Attributes;
    DWORD Priority;
    DWORD DefaultPriority;
    DWORD StartTime;
    DWORD UntilTime;
    DWORD Status;
    DWORD cJobs;
    DWORD AveragePPM;
} PRINTER_INFO_2;
```

pShareName: This member MUST be null or MUST be a non-null pointer to a string that MUST specify the share name for the printer. This string MUST be ignored unless the Attribute member in this structure contains the PRINTER_ATTRIBUTED_SHARED flag. For rules governing path names, see section [2.2.4.7](#).

pDriverName: This member MUST be a non-null pointer to a string that MUST specify the name of the printer driver. For rules governing printer driver names, see section [2.2.4.6](#).

pLocation: This member MUST be null or MUST be a non-null pointer to a string that MUST specify the location of the printer.

pDevMode: This member MUST be null or MUST be a non-null pointer to a [DEVMODE](#) structure that defines default printer data, such as the paper orientation and the resolution. For more information on the DEVMODE structure, see section [2.2.2.1](#) and [\[DEVMODE\]](#).

pSepFile: This member MUST be null or MUST be a non-null pointer to an empty string or MUST be a non-null pointer to a string that MUST specify the name of the file whose contents are used to create the separator page. This page is used to separate print jobs sent to the printer. For rules governing path names, see section [2.2.4.7](#).

pPrintProcessor: This member MUST be a non-null pointer to a string that MUST specify the name of the print processor used by the printer. For rules governing print processor names, see section [2.2.4.8](#).

pDatatype: This member MUST be null or MUST be a non-null pointer to a string that MUST specify the data type used to record the print job. For rules governing datatype names, see section [2.2.4.15](#).

pParameters: This member MUST be null or MUST be a non-null pointer to a string that MUST specify the default print processor parameters.

Priority: The value of this member MUST specify a priority value that the spooler uses to route each print job. The value of this member MUST be a decimal number from 0 through 99, inclusive.

DefaultPriority: The value of this member MUST specify the default priority value assigned to each print job. The value of this member MUST be a decimal number from 0 through 99, inclusive.

StartTime: The value of this member MUST specify the earliest time that the job can be printed. The time is expressed as the number of minutes after 12:00 AM GMT within a 24-hour boundary.

UntilTime: The value of this member MUST specify the latest time that the job can be printed. The time is expressed as the number of minutes after 12:00 AM GMT within a 24-hour boundary.

cJobs: The value of this member MUST specify the number of print jobs that have been queued for the printer.

AveragePPM: The value of this member MUST specify the average pages per minute that have been printed on the printer.

All members not defined in this section are specified in sections [2.2.1.3.6](#) and [2.2.1.3](#).

2.2.1.10.4 PRINTER_INFO_3

The **PRINTER_INFO_3** structure provides information about a printer.

```
typedef struct _PRINTER_INFO_3 {  
    SECURITY_DESCRIPTOR* pSecurityDescriptor;  
} PRINTER_INFO_3;
```

All members not defined in this section are specified in sections [2.2.1.3.6](#) and [2.2.1.3](#).

2.2.1.10.5 PRINTER_INFO_4

The **PRINTER_INFO_4** structure provides information about a printer.

```
typedef struct _PRINTER_INFO_4 {
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pServerName;
    DWORD Attributes;
} PRINTER_INFO_4;
```

All members not defined in this section are specified in sections [2.2.1.3.6](#) and [2.2.1.3](#).

2.2.1.10.6 PRINTER_INFO_5

The **PRINTER_INFO_5** structure provides information about a printer.

```
typedef struct _PRINTER_INFO_5 {
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pPortName;
    DWORD Attributes;
    DWORD DeviceNotSelectedTimeout;
    DWORD TransmissionRetryTimeout;
} PRINTER_INFO_5;
```

DeviceNotSelectedTimeout: The value of this member MUST specify the maximum number of milliseconds between select attempts.

TransmissionRetryTimeout: The value of this member MUST specify the maximum number of milliseconds between retransmission attempts.

All members not defined in this section are specified in sections [2.2.1.3.6](#) and [2.2.1.3](#).

2.2.1.10.7 PRINTER_INFO_6

The **PRINTER_INFO_6** structure provides information about a printer.

```
typedef struct _PRINTER_INFO_6 {
    DWORD dwStatus;
} PRINTER_INFO_6;
```

dwStatus: The value of this member MUST specify the printer status. It MUST be the result of a bitwise OR of zero or more of the printer status values defined in section [2.2.3.1](#).

All members not defined in this section are specified in sections [2.2.1.3.6](#) and [2.2.1.3](#).

2.2.1.10.8 PRINTER_INFO_7

The **PRINTER_INFO_7** structure provides directory services information about a printer.

```
typedef struct _PRINTER_INFO_7 {  
    [string] wchar_t* pszObjectGUID;  
    DWORD dwAction;  
} PRINTER_INFO_7;
```

pszObjectGUID: This member SHOULD be null and MUST be ignored by the server if it is used by the client in a call to [RpcSetPrinter](#), or it MUST be set by the server to a non-null pointer to a string that MUST represent the Globally Unique Identifier (GUID) used by the directory service to identify this printer if used in a response to [RpcGetPrinter](#).

The GUID string MUST conform to the Universally Unique Identifier (UUID) grammar, as specified in [\[RFC4122\]](#) section 3.[<37>](#)

dwAction: The value of this member MUST specify an action for the printer to perform if used by the client in a call to **RpcSetPrinter**. The value of this member MUST be set to a value that represents a directory service-specific publishing state by the server if it is used in a response to **RpcGetPrinter**.

The value of this member MUST be a constant from the following table.

Name	Meaning
DSPRINT_PUBLISH 0x00000001	RpcSetPrinter: Server MUST publish the printer's data in the directory service (DS). RpcGetPrinter: Server MUST set this value to indicate the printer is published in the DS.
DSPRINT_UPDATE 0x00000002	RpcSetPrinter: Server MUST update the printer's published data in the DS. RpcGetPrinter: This value MUST NOT be returned by the server.
DSPRINT_UNPUBLISH 0x00000004	RpcSetPrinter: Server MUST remove the printer's published data from the DS. RpcGetPrinter: Server MUST set this value to indicate the printer is not published.
DSPRINT_REPUBLISH 0x00000008	RpcSetPrinter: Server MUST unpublish and republish again the DS data for the printer, refreshing all properties in the published printer. Republishing also MUST change the GUID of the published printer. RpcGetPrinter: The server MUST NOT set this value.
DSPRINT_PENDING 0x80000000	RpcSetPrinter: This value MUST NOT be used by the client. RpcGetPrinter: Server MUST return this value if a previous publish or unpublish action initiated by RpcSetPrinter is still in progress.

For details on printer attribute values, see section [2.2.2.12](#).

All members not defined in this section are specified in sections [2.2.1.3.6](#) and [2.2.1.3](#).

2.2.1.10.9 PRINTER_INFO_8

The **PRINTER_INFO_8** structure provides information about a printer.

```
typedef struct _PRINTER_INFO_8 {  
    DEVMODE* pDevMode;  
} PRINTER_INFO_8;
```

pDevMode: This member MUST be a non-null pointer to a [DEVMODE](#) structure that defines global default printer data. For more information about the DEVMODE structure see section [2.2.2.1](#) and [\[DEVMODE\]](#).

All members not defined in this section are specified in sections [2.2.1.3.6](#) and [2.2.1.3](#).

2.2.1.10.10 PRINTER_INFO_9

The **PRINTER_INFO_9** structure provides information about a printer.

```
typedef struct _PRINTER_INFO_9 {  
    DEVMODE* pDevMode;  
} PRINTER_INFO_9;
```

pDevMode: This member MUST be a non-null pointer to a [DEVMODE](#) structure that defines per-user default printer data. For more information about the DEVMODE structure, see section [2.2.2.1](#) and [\[DEVMODE\]](#).

All members not defined in this section are specified in sections [2.2.1.3.6](#) and [2.2.1.3](#).

2.2.1.11 SPLCLIENT_INFO

2.2.1.11.1 SPLCLIENT_INFO_1

The **SPLCLIENT_INFO_1** structure provides information about the calling client of the print server.

```
typedef struct _SPLCLIENT_INFO_1 {  
    DWORD dwSize;  
    wchar_t* pMachineName;  
    wchar_t* pUserName;  
    DWORD dwBuildNum;  
    DWORD dwMajorVersion;  
    DWORD dwMinorVersion;  
    unsigned short wProcessorArchitecture;  
} SPLCLIENT_INFO_1;
```

dwSize: The value of this member MUST specify the size, in bytes, of the structure.

All members not defined in this section are specified in sections [2.2.1.3.7](#) and [2.2.1.3](#).

2.2.1.11.2 SPLCLIENT_INFO_3

The **SPLCLIENT_INFO_3** structure provides information about the calling client of the print server.

```
typedef struct _SPLCLIENT_INFO_3 {
    unsigned int cbSize;
    DWORD dwFlags;
    DWORD dwSize;
    [string] wchar_t* pMachineName;
    [string] wchar_t* pUserName;
    DWORD dwBuildNum;
    DWORD dwMajorVersion;
    DWORD dwMinorVersion;
    unsigned short wProcessorArchitecture;
    unsigned __int64 hSplPrinter;
} SPLCLIENT_INFO_3;
```

cbSize: The value of this member MUST specify the size, in bytes, of the structure.

dwFlags: This member is reserved for future use. The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

dwSize: This member is reserved for future use. The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

hSplPrinter: This member MUST NOT be used remotely and the value of this member SHOULD be set to zero for calls that are made remotely.

All members not defined in this section are specified in sections [2.2.1.3.7](#) and [2.2.1.3](#).

2.2.1.12 Bidirectional Communication Data

2.2.1.12.1 RPC_BIDI_REQUEST_DATA

The **RPC_BIDI_REQUEST_DATA** structure holds a single bidirectional request. The request is part of a bidirectional communication request using the [RpcSendRecvBidiData](#) method. One or more **RPC_BIDI_REQUEST_DATA** structures MUST be contained in a [RPC_BIDI_REQUEST_CONTAINER](#).

```
typedef struct _RPC_BIDI_REQUEST_DATA {
    DWORD dwReqNumber;
    [string, unique] wchar_t* pSchema;
    RPC_BIDI_DATA data;
} RPC_BIDI_REQUEST_DATA;
```

dwReqNumber: The value of this member MUST specify the index of the request, and it is used to match the response to the request in a multi-request operation.

pSchema: This member MUST be a non-null pointer to the schema string that MUST identify the requested information. [<38>](#)

data: This member MUST contain the data that is associated with the schema.

2.2.1.12.2 RPC_BIDI_RESPONSE_DATA

The **RPC_BIDI_RESPONSE_DATA** structure holds a single bidirectional response.

```
typedef struct _RPC_BIDI_RESPONSE_DATA {
    DWORD dwResult;
    DWORD dwReqNumber;
    [string, unique] wchar_t* pSchema;
    RPC_BIDI_DATA data;
} RPC_BIDI_RESPONSE_DATA;
```

dwResult: The value of this member MUST specify the result of the operation that used this structure. If the operation was successful, the value of this member MUST be set to zero; otherwise, the value of this member MUST be set to a nonzero value.[<39>](#)

dwReqNumber: The value of this member MUST specify the index of the response, and it is used to match the response to the request in a multi-request operation.

pSchema: This member MUST be a non-null pointer to the schema string that MUST identify the requested information.[<40>](#)

data: This member MUST contain the data that is associated with the schema. The data MAY be a single piece of data or a homogeneous data list. The data MUST be composed of a name, a type, and a value, and is referenced by its name under **Properties** (for example, "\Printer.Stapler.CurrentValue").

2.2.1.12.3 RPC_BIDI_DATA

The **RPC_BIDI_DATA** structure is used to store the values of a bidirectional schema.

```
typedef struct _RPC_BIDI_DATA {
    DWORD dwBidiType;
    [switch_is(dwBidiType)] union {
        [case(BIDI_NULL, BIDI_BOOL)]
            int bData;
        [case(BIDI_INT)]
            long iData;
        [case(BIDI_STRING, BIDI_TEXT, BIDI_ENUM)]
            [unique, string] wchar_t* sData;
        [case(BIDI_FLOAT)]
            float fData;
        [case(BIDI_BLOB)]
            RPC_BINARY_CONTAINER biData;
    } u;
} RPC_BIDI_DATA;
```

dwBidiType: The value of this member MUST specify the type of data in a bidirectional request. The value of this member specifies the valid structure for the **u** union. The value of this member MUST be one of the **BIDI_TYPE** enumeration values specified in section [2.2.3.10](#).

u: This member **MUST** contain the bidirectional data in the format specified by the value of the **dwBidiType** member.

bData: Boolean value.

dwBidiType member value	Meaning
BIDI_NULL 0x00000000	No data.
BIDI_BOOL 0x00000003	Bidirectional data is either 0 or 1.

iData: Integer value.

dwBidiType member value	Meaning
BIDI_INT 0x00000001	Bidirectional data is an integer.

sData: Points to a string.

dwBidiType member value	Meaning
BIDI_STRING 0x00000004	Bidirectional data is a string.
BIDI_TEXT 0x00000005	Bidirectional data is a string that SHOULD NOT be localized.<41>
BIDI_ENUM 0x00000006	Bidirectional data is enumeration data in the form of a string.

fData: Floating-point value.

dwBidiType member value	Meaning
BIDI_FLOAT 0x00000002	Bidirectional data is a floating-point number.

biData: An [RPC_BINARY_CONTAINER](#) structure that holds the binary data.

dwBidiType member value	Meaning
BIDI_BLOB	Bidirectional data is binary.

dwBidiType member value	Meaning
0x00000007	

2.2.1.13 Printer Notification Data

2.2.1.13.1 RPC_V2_NOTIFY_OPTIONS

The **RPC_V2_NOTIFY_OPTIONS** structure MUST specify options for a change notification object that monitors a printer or print server for any changes in state.

```
typedef struct _RPC_V2_NOTIFY_OPTIONS {
    DWORD Version;
    DWORD Reserved;
    DWORD Count;
    [size_is(Count), unique] RPC_V2_NOTIFY_OPTIONS_TYPE* pTypes;
} RPC_V2_NOTIFY_OPTIONS;
```

Version: The value of this member specifies the version of this structure. This value MUST be 0x02.

Reserved: (Note: Intentionally named "Reserved"). The value of this member MUST be set to the result of the bitwise OR of zero or more of the following values:

Name/Value	Meaning
PRINTER_NOTIFY_OPTIONS_REFRESH 0x00000001	The client MUST set this flag to request refreshed data from the server for all monitored members.
All_other_bits_MUST_be_zero. 0x00000000	

Count: The value of this member MUST specify the number of elements in the **pTypes** array.

pTypes: This member MUST be a non-null pointer to an array of pointers to [RPC_V2_NOTIFY_OPTIONS_TYPE](#) structures. The array MAY contain elements that specify printer information members to monitor, one element that specifies job information members to monitor, or both.

2.2.1.13.2 RPC_V2_NOTIFY_OPTIONS_TYPE

The **RPC_V2_NOTIFY_OPTIONS_TYPE** structure MUST specify the set of printer or job information members to be monitored by a printer change notification object.

```
typedef struct _RPC_V2_NOTIFY_OPTIONS_TYPE {
    unsigned short Type;
    unsigned short Reserved0;
    DWORD Reserved1;
    DWORD Reserved2;
    DWORD Count;
    [size_is(Count), unique] unsigned short* pFields;
} RPC_V2_NOTIFY_OPTIONS_TYPE;
```

Type: The value of this member specifies the type of notification to watch for. The value of this member MUST be one of the constant values from the following table:

Name/Value	Meaning
PRINTER_NOTIFY_TYPE 0x0000	Indicates that the members specified in the array that is pointed to by the pFields member are printer notification constants.
JOB_NOTIFY_TYPE 0x0001	Indicates that the members specified in the array that is pointed to by the pFields member are job notification constants.

Reserved0: The value of this member MUST be set to zero and this member MUST be ignored upon receipt.

Reserved1: The value of this member MUST be set to zero and this member MUST be ignored upon receipt.

Reserved2: The value of this member MUST be set to zero and this member MUST be ignored upon receipt.

Count: The value of this member MUST specify the number of elements in the **pFields** array.

pFields: This member MUST be a non-null pointer to an array that MUST identify the job or printer information members to be monitored. The array MUST consist entirely of elements that are either job notification values (as specified in section [2.2.3.6](#)) or printer notification values (as specified in section [2.2.3.5](#)), depending on the value of the **Type** member. The two types of notification values MUST NOT be mixed within a given instance of the array.

2.2.1.13.3 RPC_V2_NOTIFY_INFO

The **RPC_V2_NOTIFY_INFO** structure provides printer information fields and provides current data for those fields.

```
typedef struct _RPC_V2_NOTIFY_INFO {
    DWORD Version;
    DWORD Flags;
    DWORD Count;
    [size_is(Count), unique] RPC_V2_NOTIFY_INFO_DATA aData[];
} RPC_V2_NOTIFY_INFO;
```

Version: Specifies the version of the structure. The value of this member MUST be set to 0x00000002.

Flags: The value of this member is a bit flag that MUST indicate the state of the notification structure. It MUST be set to the result of a bitwise OR of zero or more of the following values:

Name/Value	Meaning
PRINTER_NOTIFY_INFO_DISCARDED 0x00000001	MUST be set by the server if an overflow or error occurred, and notifications have been lost. The server MUST NOT send

Name/Value	Meaning
	additional notifications until the client has made an RpcRouterRefreshPrinterChangeNotification call.

Count: The value of this member MUST specify the number of [RPC_V2_NOTIFY_INFO_DATA](#) elements in the **aData** array.

aData: This member is an array of **RPC_V2_NOTIFY_INFO_DATA** structures. Each element of the array MUST identify a single job or printer information member and provide the current data for that member.

2.2.1.13.4 RPC_V2_NOTIFY_INFO_DATA

The **RPC_V2_NOTIFY_INFO_DATA** structure provides a job or printer information member and provides the current data for that member.

```
typedef struct _RPC_V2_NOTIFY_INFO_DATA {
    unsigned short Type;
    unsigned short Field;
    DWORD Reserved;
    DWORD Id;
    [switch_is(Reserved & 0xffff)]
    RPC_V2_NOTIFY_INFO_DATA_DATA Data;
} RPC_V2_NOTIFY_INFO_DATA;
```

Type: The value of this member specifies the type of information that is provided in this structure. The value of this member MUST be one of the constant values from the following table.

Name/Value	Meaning
PRINTER_NOTIFY_TYPE 0x0000	Printer-related notifications
JOB_NOTIFY_TYPE 0x0001	Job-related notifications

Field: The value of this member MUST specify the member that changed using the printer notification values and job notification values in sections [2.2.3.5](#) and [2.2.3.6](#).

Reserved: The value of this member MUST indicate the member of the [RPC_V2_NOTIFY_INFO_DATA_DATA](#) union that is used to specify the data type of the **Data** member. Only the 16 least-significant bits of the reserved member MUST be used to determine the type of data in the **Data** member. The value of these bits MUST be one of the constant values from the following table.

Name/Value	Meaning
TABLE_DWORD 0x0001	The Data member MUST contain a two-DWORD array.
TABLE_STRING	The Data member MUST contain a string.

Name/Value	Meaning
0x0002	
TABLE_DEVMODE 0x0003	The Data member MUST be a non-null pointer to a DEVMODE structure.
TABLE_TIME 0x0004	The Data member MUST contain a SYSTEMTIME_CONTAINER structure, as specified in section 2.2.1.2.16 .
TABLE_SECURITYDESCRIPTOR 0x0005	The Data member MUST contain a SECURITY_CONTAINER structure.

Id: The value of this member MUST contain the job identifier if the **Type** member specifies JOB_NOTIFY_TYPE, 0x01; otherwise, if the **Type** member specifies PRINTER_NOTIFY_TYPE, 0x00, the value of this member MUST be ignored.

Data: This member MUST contain the data determined by the values of the **Type** and **Member** members of this structure. The data is in an **RPC_V2_NOTIFY_INFO_DATA_DATA** structure using the data type described by the value of the **Reserved** member.

2.2.1.13.5 RPC_V2_NOTIFY_INFO_DATA_DATA

The **RPC_V2_NOTIFY_INFO_DATA_DATA** union defines the data information container for the current notification. The numeric values of the case attributes of this union are specified in section [2.2.1.13.4](#).

```
typedef
[switch_type (DWORD)]
union _RPC_V2_NOTIFY_INFO_DATA_DATA {
    [case(TABLE_STRING)]
        STRING_CONTAINER String;
    [case(TABLE_DWORD)] DWORD dwData[2];
    [case(TABLE_TIME)] SYSTEMTIME_CONTAINER SystemTime;
    [case(TABLE_DEVMODE)] DEVMODE_CONTAINER DevMode;
    [case(TABLE_SECURITYDESCRIPTOR)]
        SECURITY_CONTAINER SecurityDescriptor;
} RPC_V2_NOTIFY_INFO_DATA_DATA;
```

String: Case **TABLE_STRING:** This member MUST specify a **STRING_CONTAINER** structure, as specified in section [2.2.1.2.15](#)

dwData: Case **TABLE_DWORD:** This member is an array of two **DWORD** values that MUST contain the member's current data.

SystemTime: Case **TABLE_TIME** This member MUST specify a **SYSTEMTIME_CONTAINER** structure, as specified in section [2.2.1.2.16](#).

DevMode: Case **TABLE_DEVMODE:** This member is a **DEVMODE_CONTAINER** structure that MUST define default printer data (for example, the paper orientation and the resolution). The **DEVMODE_CONTAINER** is specified in section [2.2.1.2.1](#).

SecurityDescriptor: Case **TABLE_SECURITYDESCRIPTOR:** This member MUST specify a **SECURITY_CONTAINER** structure in which the **pSecurity** member MUST either be null or a non-null pointer to a [SECURITY_DESCRIPTOR](#) structure in self-relative form. The

SECURITY_CONTAINER is specified in section [2.2.1.2.13](#); the **SECURITY_DESCRIPTOR** is specified in [\[MS-DTYP\]](#) section **2.4.6**.

2.2.1.13.6 RPC_V2_UREPLY_PRINTER

The **RPC_V2_UREPLY_PRINTER** union defines printer notification responses.

```
typedef
[switch_type (DWORD)]
union _RPC_V2_UREPLY_PRINTER {
    [case (0)] RPC_V2_NOTIFY_INFO* pInfo;
} RPC_V2_UREPLY_PRINTER;
```

pInfo: This member MUST be a non-null pointer to an **RPC_V2_NOTIFY_INFO** structure, which MUST contain notification information.

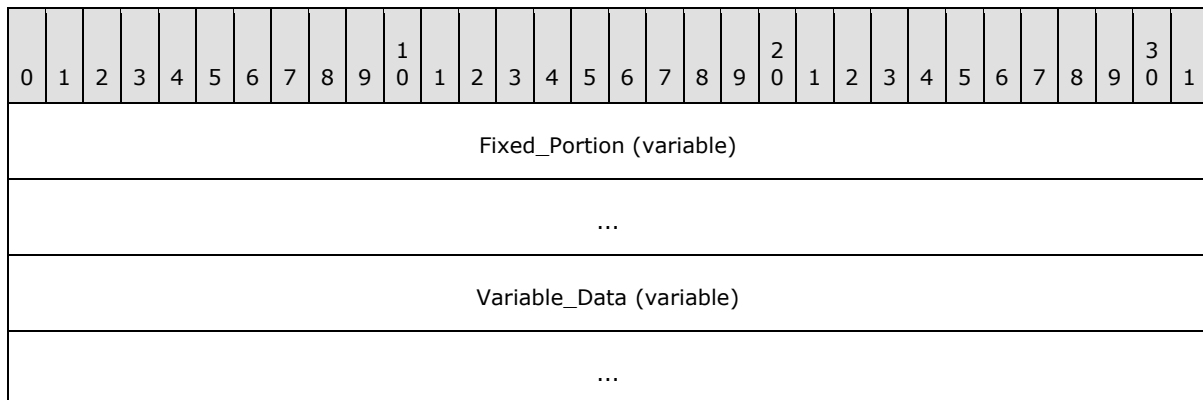
2.2.2 Custom-Marshaled Data Types

Data structures containing "_INFO" in their names are custom-marshaled as single **fixed-portion** blocks for methods accepting or returning a single structure, and are custom-marshaled as arrays of one or more **fixed-portion** blocks for methods accepting or returning an array of structures.

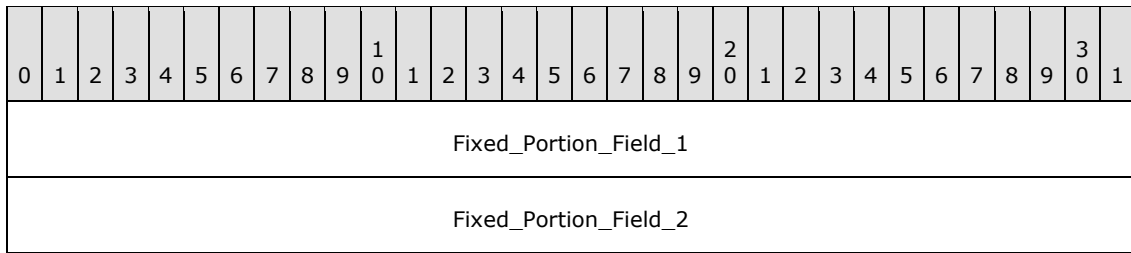
With the exception of [ADDJOB_INFO_1](#) as detailed in the [RpcAddJob](#) method, all custom-marshaled INFO structures MUST be completely ignored on input, and validation of their contents MUST NOT take place.

The one or more **fixed-portion** blocks are followed by a single **variable-data** area, which contains variable-length string or multisz data members, located by adding the corresponding **Offset** member value to the offset of the start of the **fixed-portion** block in which the **Offset** member resides.

This generic packet structure can be represented by the following diagram:



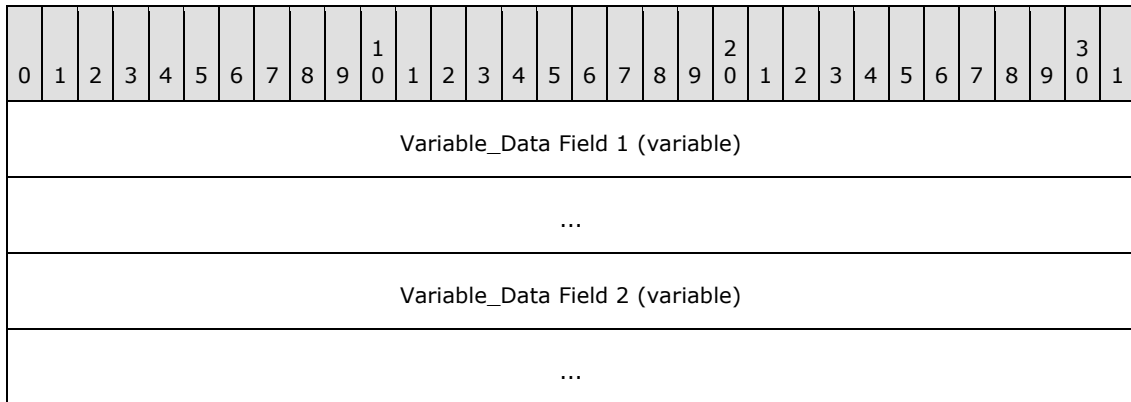
Fixed_Portion (variable): An array of one or more **fixed-portion** blocks, the specific structure of which is defined for each "_INFO" structure.



Fixed_Portion_Field_1 (4 bytes): This is **fixed-portion** field 1.

Fixed_Portion_Field_2 (4 bytes): This is **fixed-portion** field 2.

Variable_Data (variable): A **variable-data** area of variable length. Because the data is not necessarily aligned on 16-bit boundaries, it is specified as an array of bytes of arbitrary length.



Variable_Data Field 1 (variable): This is **variable-data** field 1.

Variable_Data Field 2 (variable): This is **variable-data** field 2.

The string or multisz data members in the **variable-data** area MAY appear in arbitrary order.

One or more **Offset** members MAY locate the same string or multisz data members, or one string or multisz data member MAY be present for each **Offset** member of each **fixed-portion** block.

The string or multisz data members SHOULD be packed tightly in the **variable-data** area. Code that processes a custom-marshaled INFO structure MUST be prepared to correctly handle data that is not tightly packed and includes unused space.

2.2.2.1 DEVMODE

The DEVMODE structure defines initialization data for a printer. Although the DEVMODE structure does not contain any pointers, it is still custom marshaled because the overall size of the structure is version-specific and implementation-specific and cannot be expressed through IDL attributes. It uses the following message format:

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
dmDeviceName																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															
(dmDeviceName cont'd for 8 rows)																															
dmSpecVersion																dmDriverVersion															
dmSize																dmDriverExtra															
dmFields																															
dmOrientation																dmPaperSize															
dmPaperLength																dmPaperWidth															
dmScale																dmCopies															
dmDefaultSource																dmPrintQuality															
dmColor																dmDuplex															
dmYResolution																dmTTOption															
dmCollate																dmFormName															

...	
...	
...	
...	
...	
...	
...	
(dmFormName cont'd for 8 rows)	
...	reserved0
reserved1	
reserved2	
reserved3	
dmNup	
reserved4	
dmICMMethod	
dmICMIntent	
dmMediaType	
dmDitherType	
reserved5	
reserved6	
reserved7	

reserved8
dmDriverExtraData (variable)
...

dmDeviceName (64 bytes): This member is a 32-element array of 16-bit **Unicode** characters that MUST specify the name of the printer. This member is restricted to 32 characters, including the trailing null. Printer names that are longer than that MUST be truncated to fit the array. For more rules governing printer names, see section [2.2.4.2](#).

dmSpecVersion (2 bytes): The value of this member MUST specify the version of initialization data specification on which the DEVMODE structure is based. It SHOULD be 0x0401. [<42>](#)

dmDriverVersion (2 bytes): For printers, the value of this member SHOULD specify the implementation-defined version of the printer driver. [<43>](#) For displays, this value MAY be the same as the **dmSpecVersion** field.

dmSize (2 bytes): The value of this member MUST specify the size, in bytes, of the DEVMODE structure. The size MUST NOT include the length of any private, printer driver-specific data that might follow the DEVMODE structure's public members.

dmDriverExtra (2 bytes): The value of this member MUST contain the size in bytes of the private printer driver data that follows this structure.

dmFields (4 bytes): The value of this member MUST specify the members of the DEVMODE structure that have been initialized. If a bit is set, the corresponding member MUST be initialized, and it MUST be processed on receipt; otherwise, if the bit is not set, the value of the corresponding member SHOULD be set to zero, and it MUST be ignored on receipt. The value of this member MUST be the result of a bitwise OR of the following bits:

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
X	U	X	S	P	P	P	O	C	T	Y	D	C	P	D	C	C	X	X	X	X	X	X	F	X	X	X	X	X	D	M	C
P	P	C	W	L	S	R	L	T		X	R	Q	S	P	M							M						T	T	I	

Value	Description
CI DM_ICMINTENT	If this bit is set, the dmICMIntent member MUST be initialized.
MT DM_MEDIATYPE	If this bit is set, the dmMediaType member MUST be initialized.
DT DM_DITHERTYPE	If this bit is set, the dmDitherType member MUST be initialized.
X	This bit MUST be clear (set to zero).

Value	Description
ReservedA	
X Reserved9	This bit MUST be clear (set to zero).
X ReservedB	This bit MUST be clear (set to zero).
X ReservedC	This bit MUST be clear (set to zero).
X ReservedD	This bit MUST be clear (set to zero).
FM DM_FORMNAME	If this bit is set, the dmFormName member MUST be initialized.
X Reserved3	This bit MUST be clear (set to zero).
X Reserved4	This bit MUST be clear (set to zero).
X Reserved5	This bit MUST be clear (set to zero).
X Reserved6	This bit MUST be clear (set to zero).
X Reserved7	This bit MUST be clear (set to zero).
X Reserved8	This bit MUST be clear (set to zero).
CM DM_ICMMETHOD	If this bit is set, the dmICMMethod member MUST be initialized.
CP DM_COPIES	If this bit is set, the dmCopies member MUST be initialized.
DS DM_DEFAULTSOURCE	If this bit is set, the dmDefaultSource member MUST be initialized.
PQ DM_PRINTQUALITY	If this bit is set, the dmPrintQuality member MUST be initialized.
CR DM_COLOR	If this bit is set, the dmColor member MUST be initialized.
DX DM_DUPLEX	If this bit is set, the dmDuplex member MUST be initialized.

Value	Description
Y DM_YRESOLUTION	If this bit is set, the dmYResolution member MUST be initialized.
TT DM_TTOPTION	If this bit is set, the dmTTOption member MUST be initialized.
CL DM_COLLATE	If this bit is set, the dmCollate member MUST be initialized.
OR DM_ORIENTATION	If this bit is set, the dmOrientation member MUST be initialized.
PS DM_PAPERSIZE	If this bit is set, the dmPaperSize member MUST be initialized. This bit MUST NOT be set if either DM_PAPERLENGTH or DM_PAPERWIDTH are set.
PL DM_PAPERLENGTH	If this bit is set, the dmPaperLength member MUST be initialized. This bit MUST NOT be set if DM_PAPERSIZE is set.
PW DM_PAPERWIDTH	If this bit is set, the dmPaperWidth member MUST be initialized. This bit MUST NOT be set if DM_PAPERSIZE is set.
SC DM_SCALE	If this bit is set, the dmScale member MUST be initialized.
X Reserved1	This bit MUST be clear (set to zero).
UP DM_NUP	If this bit is set, the dmNup member MUST be initialized.
X Reserved2	This bit MUST be clear (set to zero).

dmOrientation (2 bytes): If the DM_ORIENTATION bit is set in the **dmFields** member, the value of this member MUST specify the orientation of the paper. The value of this member MUST be either one, for portrait orientation, or two, for landscape orientation.

dmPaperSize (2 bytes): If the DM_PAPERSIZE bit is set in the **dmFields** member, the value of this member MUST specify the size of the paper to print on. The value of this member MUST be one of the constant values from the following table, or it SHOULD be a device-specific value that is greater than or equal to 0x0100:

Name/Value	Meaning
DMPAPER_LETTER 0x0001	Letter, 8 1/2 x 11 inches
DMPAPER_LEGAL 0x0005	Legal, 8 1/2 x 14 inches
DMPAPER_10X14 0x0010	10 x 14-inch sheet

Name/Value	Meaning
DMPAPER_11X17 0x0011	11 x 17-inch sheet
DMPAPER_12X11 0x005A	Windows Me, Windows 98, Windows NT 4.0 and later: 12 x 11-inch sheet
DMPAPER_A3 0x0008	A3 sheet, 297 x 420 millimeters
DMPAPER_A3_ROTATED 0x004C	Windows Me, Windows 98, Windows NT 4.0 and later: A3 rotated sheet, 420 x 297 millimeters
DMPAPER_A4 0x0009	A4 sheet, 210 x 297 millimeters
DMPAPER_A4_ROTATED 0x004D	Windows Me, Windows 98, Windows NT 4.0 and later: A4 rotated sheet, 297 x 210 millimeters
DMPAPER_A4SMALL 0x000A	A4 small sheet, 210 x 297 millimeters
DMPAPER_A5 0x000B	A5 sheet, 148 x 210 millimeters
DMPAPER_A5_ROTATED 0x004E	Windows Me, Windows 98, Windows NT 4.0 and later: A5 rotated sheet, 210 x 148 millimeters
DMPAPER_A6 0x0046	Windows Me, Windows 98, Windows NT 4.0 and later: A6 sheet, 105 x 148 millimeters
DMPAPER_A6_ROTATED 0x0053	Windows Me, Windows 98, Windows NT 4.0 and later: A6 rotated sheet, 148 x 105 millimeters
DMPAPER_B4 0x000C	B4 sheet, 250 x 354 millimeters
DMPAPER_B4_JIS_ROTATED 0x004F	Windows Me, Windows 98, Windows NT 4.0 and later: B4 (JIS) rotated sheet, 364 x 257 millimeters
DMPAPER_B5 0x000D	B5 sheet, 182 x 257-millimeter paper
DMPAPER_B5_JIS_ROTATED 0x0050	Windows Me, Windows 98, Windows NT 4.0 and later: B5 (JIS) rotated sheet, 257 x 182 millimeters
DMPAPER_B6_JIS 0x0058	Windows Me, Windows 98, Windows NT 4.0 and later: B6 (JIS) sheet, 128 x 182 millimeters
DMPAPER_B6_JIS_ROTATED 0x0059	Windows Me, Windows 98, Windows NT 4.0 and later: B6 (JIS) rotated sheet, 182 x 128 millimeters
DMPAPER_CSHEET 0x0018	C Sheet, 17 x 22 inches
DMPAPER_DBL_JAPANESE_POSTCARD	Windows Me, Windows 98, Windows NT 4.0 and

Name/Value	Meaning
0x0045	later: Double Japanese Postcard, 200 x 148 millimeters
DMPAPER_DBL_JAPANESE_POSTCARD_ROTATED 0x0052	Windows 98, Windows Me, Windows NT 4.0 and later: Double Japanese Postcard Rotated, 148 x 200 millimeters
DMPAPER_DSHEET 0x0019	D Sheet, 22 x 34 inches
DMPAPER_ENV_9 0x0013	#9 Envelope, 3 7/8 x 8 7/8 inches
DMPAPER_ENV_10 0x0014	#10 Envelope, 4 1/8 x 9 1/2 inches
DMPAPER_ENV_11 0x0015	#11 Envelope, 4 1/2 x 10 3/8 inches
DMPAPER_ENV_12 0x0016	#12 Envelope, 4 3/4 x 11 inches
DMPAPER_ENV_14 0x0017	#14 Envelope, 5 x 11 1/2 inches
DMPAPER_ENV_C5 0x001C	C5 Envelope, 162 x 229 millimeters
DMPAPER_ENV_C3 0x001D	C3 Envelope, 324 x 458 millimeters
DMPAPER_ENV_C4 0x001E	C4 Envelope, 229 x 324 millimeters
DMPAPER_ENV_C6 0x001F	C6 Envelope, 114 x 162 millimeters
DMPAPER_ENV_C65 0x0020	C65 Envelope, 114 x 229 millimeters
DMPAPER_ENV_B4 0x0021	B4 Envelope, 250 x 353 millimeters
DMPAPER_ENV_B5 0x0022	B5 Envelope, 176 x 250 millimeters
DMPAPER_ENV_B6 0x0023	B6 Envelope, 176 x 125 millimeters
DMPAPER_ENV_DL 0x001B	DL Envelope, 110 x 220 millimeters
DMPAPER_ENV_ITALY 0x0024	Italy Envelope, 110 x 230 millimeters
DMPAPER_ENV_MONARCH 0x0025	Monarch Envelope, 3 7/8 x 7 1/2 inches

Name/Value	Meaning
DMPAPER_ENV_PERSONAL 0x0026	6 3/4 Envelope, 3 5/8 x 6 1/2 inches
DMPAPER_ESHEET 0x001A	E Sheet, 34 x 44 inches
DMPAPER_EXECUTIVE 0x0007	Executive, 7 1/4 x 10 1/2 inches
DMPAPER_FANFOLD_US 0x0027	US Std Fanfold, 14 7/8 x 11 inches
DMPAPER_FANFOLD_STD_GERMAN 0x0028	German Std Fanfold, 8 1/2 x 12 inches
DMPAPER_FANFOLD_LGL_GERMAN 0x0029	German Legal Fanfold, 8 x 13 inches
DMPAPER_FOLIO 0x000E	Folio, 8 1/2 x 13-inch paper
DMPAPER_JAPANESE_POSTCARD_ROTATED 0x0051	Windows Me, Windows 98, Windows NT 4.0 and later: Japanese Postcard Rotated, 148 x 100 millimeters
DMPAPER_JENV_CHOU3 0x0049	Windows Me, Windows 98, Windows NT 4.0 and later: Japanese Envelope Chou #3
DMPAPER_JENV_CHOU3_ROTATED 0x0056	Windows Me, Windows 98, Windows NT 4.0 and later: Japanese Envelope Chou #3 Rotated
DMPAPER_JENV_CHOU4 0x004A	Windows Me, Windows 98, Windows NT 4.0 and later: Japanese Envelope Chou #4
DMPAPER_JENV_CHOU4_ROTATED 0x0057	Windows Me, Windows 98, Windows NT 4.0 and later: Japanese Envelope Chou #4 Rotated
DMPAPER_JENV_KAKU2 0x0047	Windows Me, Windows 98, Windows NT 4.0 and later: Japanese Envelope Kaku #2
DMPAPER_JENV_KAKU2_ROTATED 0x0054	Windows Me, Windows 98, Windows NT 4.0 and later: Japanese Envelope Kaku #2 Rotated
DMPAPER_JENV_KAKU3 0x0048	Windows Me, Windows 98, Windows NT 4.0 and later: Japanese Envelope Kaku #3
DMPAPER_JENV_KAKU3_ROTATED 0x0055	Windows Me, Windows 98, Windows NT 4.0 and later: Japanese Envelope Kaku #3 Rotated
DMPAPER_JENV_YOU4 0x005B	Windows Me, Windows 98, Windows NT 4.0 and later: Japanese Envelope You #4
DMPAPER_JENV_YOU4_ROTATED 0x005C	Windows Me, Windows 98, Windows NT 4.0 and later: Japanese Envelope You #4
DMPAPER_LEDGER 0x0004	Ledger, 17 x 11 inches

Name/Value	Meaning
DMPAPER_LETTER_ROTATED 0x004B	Letter Rotated, 11 by 8 1/2 inches
DMPAPER_LETTERSMALL 0x0002	Letter Small, 8 1/2 x 11 inches
DMPAPER_NOTE 0x0012	Note, 8 1/2 x 11-inches
DMPAPER_P16K 0x005D	Windows Me, Windows 98, Windows NT 4.0 and later: PRC 16K, 146 x 215 millimeters
DMPAPER_P16K_ROTATED 0x006A	Windows Me, Windows 98, Windows NT 4.0 and later: PRC 16K Rotated, 215 x 146 millimeters
DMPAPER_P32K 0x005E	Windows Me, Windows 98, Windows NT 4.0 and later: PRC 32K, 97 x 151 millimeters
DMPAPER_P32K_ROTATED 0x006B	Windows Me, Windows 98, Windows NT 4.0 and later: PRC 32K Rotated, 151 x 97 millimeters
DMPAPER_P32KBIG 0x005F	Windows Me, Windows 98, Windows NT 4.0 and later: PRC 32K(Big) 97 x 151 millimeters
DMPAPER_P32KBIG_ROTATED 0x006C	Windows Me, Windows 98, Windows NT 4.0 and later: PRC 32K(Big) Rotated, 151 x 97 millimeters
DMPAPER_PENV_1 0x0060	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #1, 102 by 165 millimeters
DMPAPER_PENV_1_ROTATED 0x006D	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #1 Rotated, 165 x 102 millimeters
DMPAPER_PENV_2 0x0061	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #2, 102 x 176 millimeters
DMPAPER_PENV_2_ROTATED 0x006E	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #2 Rotated, 176 x 102 millimeters
DMPAPER_PENV_3 0x0062	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #3, 125 x 176 millimeters
DMPAPER_PENV_3_ROTATED 0x006F	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #3 Rotated, 176 x 125 millimeters
DMPAPER_PENV_4 0x0063	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #4, 110 x 208 millimeters
DMPAPER_PENV_4_ROTATED 0x0070	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #4 Rotated, 208 x 110 millimeters
DMPAPER_PENV_5 0x0064	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #5, 110 x 220 millimeters

Name/Value	Meaning
DMPAPER_PENV_5_ROTATED 0x0071	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #5 Rotated, 220 x 110 millimeters
DMPAPER_PENV_6 0x0065	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #6, 120 x 230 millimeters
DMPAPER_PENV_6_ROTATED 0x0072	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #6 Rotated, 230 x 120 millimeters
DMPAPER_PENV_7 0x0066	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #7, 160 x 230 millimeters
DMPAPER_PENV_7_ROTATED 0x0073	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #7 Rotated, 230 x 160 millimeters
DMPAPER_PENV_8 0x0067	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #8, 120 x 309 millimeters
DMPAPER_PENV_8_ROTATED 0x0074	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #8 Rotated, 309 x 120 millimeters
DMPAPER_PENV_9 0x0068	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #9, 229 x 324 millimeters
DMPAPER_PENV_9_ROTATED 0x0075	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #9 Rotated, 324 x 229 millimeters
DMPAPER_PENV_10 0x0069	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #10, 324 x 458 millimeters
DMPAPER_PENV_10_ROTATED 0x0076	Windows Me, Windows 98, Windows NT 4.0 and later: PRC Envelope #10 Rotated, 458 x 324 millimeters
DMPAPER_QUARTO 0x000F	Quarto, 215 x 275 millimeter paper
DMPAPER_STATEMENT 0x0006	Statement, 5 1/2 x 8 1/2 inches
DMPAPER_TABLOID 0x0003	Tabloid, 11 x 17 inches

dmPaperLength (2 bytes): If the DM_PAPERLENGTH bit is set in the **dmFields** member, the value of this member MUST specify the length of the paper, in tenths of a millimeter, to use in the printer for which the job is destined.

dmPaperWidth (2 bytes): If the DM_PAPERWIDTH bit is set in the **dmFields** member, the value of this member MUST specify the width of the paper, in tenths of a millimeter, to use in the printer for which the job is destined.

dmScale (2 bytes): If the DM_SCALE bit is set in the **dmFields** member, the value of this member MUST specify the percentage factor by which the printed output is to be scaled.

dmCopies (2 bytes): If the DM_COPIES bit is set in the **dmFields** member, the value of this member MUST provide the number of copies printed, if the device supports multiple-page copies.

dmDefaultSource (2 bytes): If the DM_DEFAULTSOURCE bit is set in the **dmFields** member, the value of this member MUST specify the paper source.

The value of this member SHOULD be one of the following, or it MAY be a device-specific value that is greater than or equal to 0x0100:

Name/Value	Meaning
DMBIN_UPPER 0x0001	Select the upper paper bin. This value is also used for the paper source for printers that only have one paper bin.
DMBIN_LOWER 0x0002	Select the lower bin.
DMBIN_MIDDLE 0x0003	Select the middle paper bin.
DMBIN_MANUAL 0x0004	Manually select the paper bin.
DMBIN_ENVELOPE 0x0005	Select the auto envelope bin.
DMBIN_ENVMANUAL 0x0006	Select the manual envelope bin.
DMBIN_AUTO 0x0007	Auto-select the bin.
DMBIN_TRACTOR 0x0008	Select the bin with the tractor paper.
DMBIN_SMALLFMT 0x0009	Select the bin with the smaller paper format.
DMBIN_LARGEFORMAT 0x000A	Select the bin with the larger paper format.
DMBIN_LARGECAPACITY 0x000B	Select the bin with large capacity.
DMBIN_CASSETTE 0x000E	Select the cassette bin.
DMBIN_FORMSOURCE 0x000F	Select the bin with the required form.

dmPrintQuality (2 bytes): If the DM_PRINTQUALITY bit is set in the **dmFields** member, the value of this member MUST specify the printer resolution. The value of this member MUST be either a positive value that specifies a device-dependent resolution in dots per inch (DPI) or one of the following four predefined device-independent values that are mapped to a device-specific resolution in an implementation-specific manner:

Name/Value	Meaning
DMRES_HIGH 0xFFFC	High-resolution printouts
DMRES_MEDIUM 0xFFFD	Medium-resolution printouts
DMRES_LOW 0xFFFE	Low-resolution printouts
DMRES_DRAFT 0xFFFF	Draft-resolution printouts

dmColor (2 bytes): If the DM_COLOR bit is set in the **dmFields** member, the value of this member MUST specify the color mode to use with color printers. The value of this member MUST be one of the following:

Name/Value	Meaning
DMRES_MONOCHROME 0x0001	Use monochrome printing mode.
DMRES_COLOR 0x0002	Use color printing mode.

dmDuplex (2 bytes): If the DM_DUPLEX bit is set in the **dmFields** member, the value of this member MUST specify duplex or double-sided printing for printers that are capable of duplex printing. The value of this member MUST be one of the following:

Name/Value	Meaning
DMDUP_SIMPLEX 0x0001	Normal (non-duplex) printing.
DMDUP_VERTICAL 0x0002	Long-edge binding, that is, the long edge of the page is vertical.
DMDUP_HORIZONTAL 0x0003	Short-edge binding, that is, the long edge of the page is horizontal.

dmYResolution (2 bytes): If the DM_YRESOLUTION bit is set in the **dmFields**, the value of this member MUST specify the y-resolution, in dots per inch, of the printer.

dmTTOption (2 bytes): If the DM_TTOPTION bit is set in the **dmFields** member, the value of this member MUST specify how TrueType fonts MUST be printed. The value of this member MUST be one of the following:

Name/Value	Meaning
DMTT_BITMAP 0x0001	Prints TrueType fonts as graphics. This is the default action for dot-matrix printers.
DMTT_DOWNLOAD 0x0002	Downloads TrueType fonts as soft fonts. This is the default action for Hewlett-Packard printers that use Printer Command Language (PCL).

Name/Value	Meaning
DMTT_SUBDEV 0x0003	Substitutes device fonts for TrueType fonts. This is the default action for PostScript printers.
DMTT_DOWNLOAD_OUTLINE 0x0004	Windows Me, Windows 98, Windows 95, Windows NT 4.0 and later: Downloads TrueType fonts as outline soft fonts.

dmCollate (2 bytes): If the DM_COLLATE bit is set in the **dmFields** member, the value of this member MUST specify whether collation MUST be used when printing multiple copies. The value of this member MUST be one of the following:

Value	Meaning
DMCOLLATE_FALSE 0x0000	Do not collate when printing multiple copies.
DMCOLLATE_TRUE 0x0001	Collate when printing multiple copies.

dmFormName (64 bytes): This member is a 32-element array of 16-bit Unicode characters. If the DM_FORMNAME bit is set in the **dmFields** member, the value of this member MUST specify the name of the form to use, for example, "Letter" or "Legal". The value of this member is restricted to 32 characters, including the trailing null. Form names that are longer than 32 characters, including the trailing null, MUST be truncated to fit the array.

reserved0 (2 bytes): The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

reserved1 (4 bytes): The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

reserved2 (4 bytes): The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

reserved3 (4 bytes): The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

dmNup (4 bytes): If the DM_NUP bit is set in the **dmFields**, the value of this member MUST specify the responsibility for performing page layout for **N-Up Printing**. It MUST be one of the following values:

Name/Value	Meaning
DMNUP_SYSTEM 0x00000001	The print server does the page layout.
DMNUP_ONEUP 0x00000002	The application does the page layout.

reserved4 (4 bytes): The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

dmICMMethod (4 bytes): If the DM_ICMMETHOD bit is set in the **dmFields** member, the value of this member MUST specify how **Image Color Management (ICM)** is handled. For a non-ICM application, this member determines if ICM is enabled or disabled. For ICM applications,

the system examines this member to determine how to handle ICM support. The value of this member **MUST** be one of the following predefined values or a printer driver-defined value greater than or equal to the value of 0x00000100.

Name/Value	Meaning
DMICMMETHOD_NONE 0x00000001	Specifies that ICM is disabled.
DMICMMETHOD_SYSTEM 0x00000002	Specifies that ICM is handled by the system on which the Page Description Language (PDL) data is generated.
DMICMMETHOD_DRIVER 0x00000003	Specifies that ICM is handled by the printer driver.
DMICMMETHOD_DEVICE 0x00000004	Specifies that ICM is handled by the destination device.

dmICMIntent (4 bytes): If the DM_ICMINTENT bit is set in the **dmFields** member, the value of this member **MUST** specify which **color matching** method, or intent, **MUST** be used by default. This member is primarily for non-ICM applications. ICM applications can establish intents by using the ICM functions. The value of this member **MUST** be one of the following predefined values, or a printer driver defined value greater than or equal to the value of 0x00000100.

Name/Value	Meaning
DMICM_SATURATE 0x00000001	Color matching SHOULD optimize for color saturation.
DMICM_CONTRAST 0x00000002	Color matching SHOULD optimize for color contrast.
DMICM_COLORIMETRIC 0x00000003	Color matching SHOULD optimize to match the exact color requested.
DMICM_ABS_COLORIMETRIC 0x00000004	Color matching SHOULD optimize to match the exact color requested without white point mapping.

dmMediaType (4 bytes): If the DM_MEDIATYPE bit is set in the **dmFields** member, the value of this member **MUST** specify the type of media to print on. The value of this member **MUST** be one of the following predefined values or else a printer driver-defined value greater than or equal to the value of 0x00000100.

Name/Value	Meaning
DMMEDIA_STANDARD 0x00000001	Plain paper
DMMEDIA_TRANSPARENCY 0x00000002	Transparent film
DMMEDIA_GLOSSY 0x00000003	Glossy paper

dmDitherType (4 bytes): If the DM_DITHERTYPE bit is set in the **dmFields** member, the value of this member **MUST** specify how **dithering** is to be done. The value of this member **MUST** be

one of the following predefined values or else a printer driver-defined value greater than or equal to 0x00000100.

Name/Value	Meaning
DMDITHER_NONE 0x00000001	No dithering.
DMDITHER_COARSE 0x00000002	Dithering with a coarse brush.
DMDITHER_FINE 0x00000003	Dithering with a fine brush.
DMDITHER_LINEART 0x00000004	Line art dithering, a special dithering method that produces well defined borders between black, white, and gray scaling.
DMDITHER_ERRORDIFFUSION 0x00000005	Windows 95/98/ME: Error diffusion dithering.
DMDITHER_RESERVED6 0x00000006	Same as DMDITHER_LINEART.
DMDITHER_RESERVED7 0x00000007	Same as DMDITHER_LINEART.
DMDITHER_RESERVED8 0x00000008	Same as DMDITHER_LINEART.
DMDITHER_RESERVED9 0x00000009	Same as DMDITHER_LINEART.
DMDITHER_GRAYSCALE 0x0000000A	Device does gray scaling.

reserved5 (4 bytes): The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

reserved6 (4 bytes): The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

reserved7 (4 bytes): The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

reserved8 (4 bytes): The value of this member SHOULD be set to zero and MUST be ignored upon receipt.

dmDriverExtraData (variable): This member MAY contain implementation-specific printer driver data. Its size in bytes is specified by the value of the **dmDriverExtra** member. [<44>](#)

2.2.2.1.1 PostScript Driver Extra Data

Information about **PostScript Driver Extra Data** can be found in [Appendix B: Windows Behavior.<45>](#)

2.2.2.1.2 Generic Driver Extra Data

Information about generic **Driver Extra Data** can be found in [Appendix B: Windows Behavior.<46>](#)

2.2.2.1.3 OEM Driver Extra Data

Information about vendor-supplied **Driver Extra Data** can be found in [Appendix B: Windows Behavior.<47>](#)

2.2.2.1.4 Print Ticket Driver Extra Data

Information about **Print Ticket Driver Extra Data** can be found in [Appendix B: Windows Behavior.<48>](#)

2.2.2.2 Members in Custom-Marshaled INFO structures

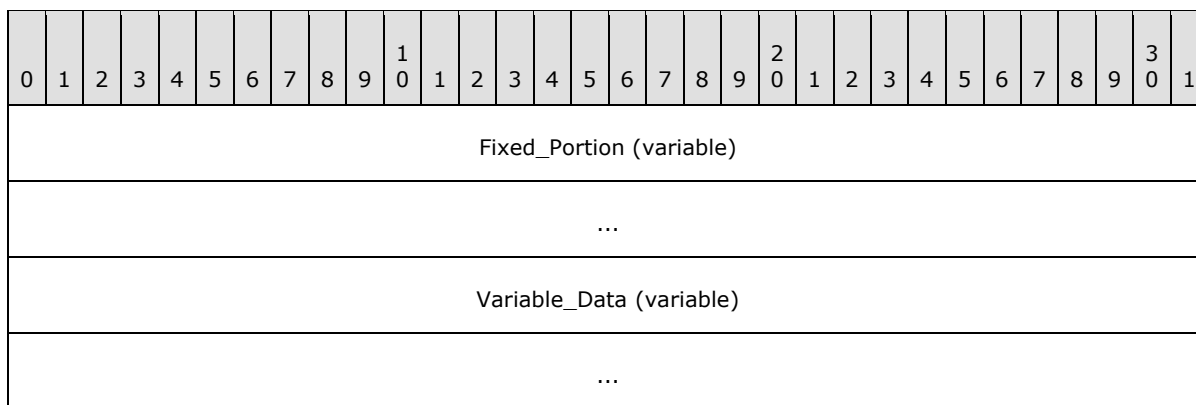
Unless noted otherwise, pointer and string pointer members of IDL-marshaled [INFO](#) structure forms are represented in the corresponding custom-marshaled INFO structure forms according to the following rules:

- **MEMBERNAME** is derived by trimming the leading "p" from the pointer, string pointer, or multisized pointer member of the IDL-marshaled INFO structure.
- The custom-marshaled INFO structure form contains an **Offset** member whose name is derived by appending "Offset" to **MEMBERNAME**.
- The **Offset** member MUST be a 32-bit unsigned integer that MUST specify the number of bytes from the start of the structure to the start of the bytes making up the pointed-to data, string, or multisized. That data, string, or multisized area is represented in the custom-marshaled structure by a member whose name is derived by appending "Array" to **MEMBERNAME**. The length of that member is variable and includes the terminating null character for string data or the two terminating null characters for multisized data, respectively.
- If the pointer, string pointer, or multisized pointer member in the IDL-marshaled structure form is optional (for example, can be null), it MAY be represented by a zero **Offset** in a custom-marshaled structure. The corresponding **Array** member is then considered optional and is present only if the **Offset** is not zero.

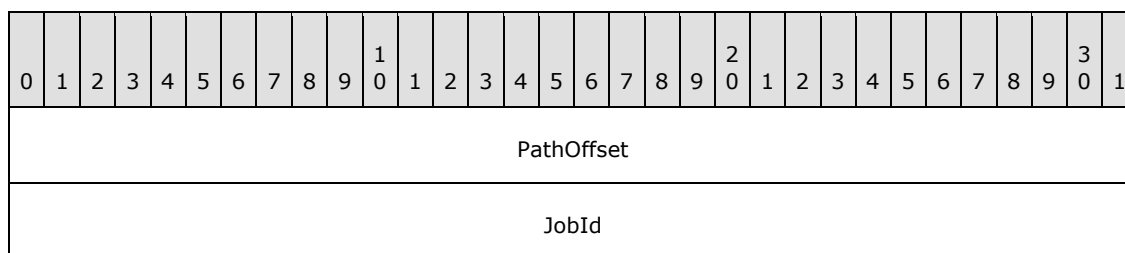
Unless noted otherwise, all other members of IDL-marshaled INFO structure forms are represented identically in the corresponding custom-marshaled INFO structure forms.

2.2.2.3 ADDJOB_INFO_1

The ADDJOB_INFO_1 structure specifies the information needed to add a file to a print job.



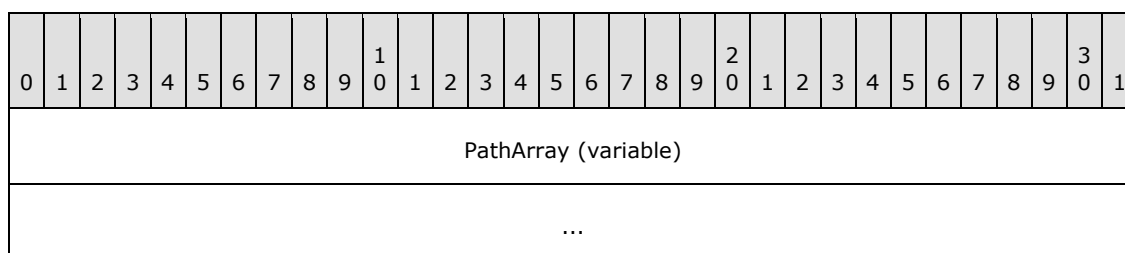
Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.



PathOffset (4 bytes): This member is a 32-bit unsigned integer that **MUST** specify the number of bytes from the start of the structure to the start of the **PathArray** member. The value of this member **MUST** be greater than or equal to the length of the fixed portion of this structure.

JobId (4 bytes): The value of this member **MUST** specify a nonzero identifier for the print job.

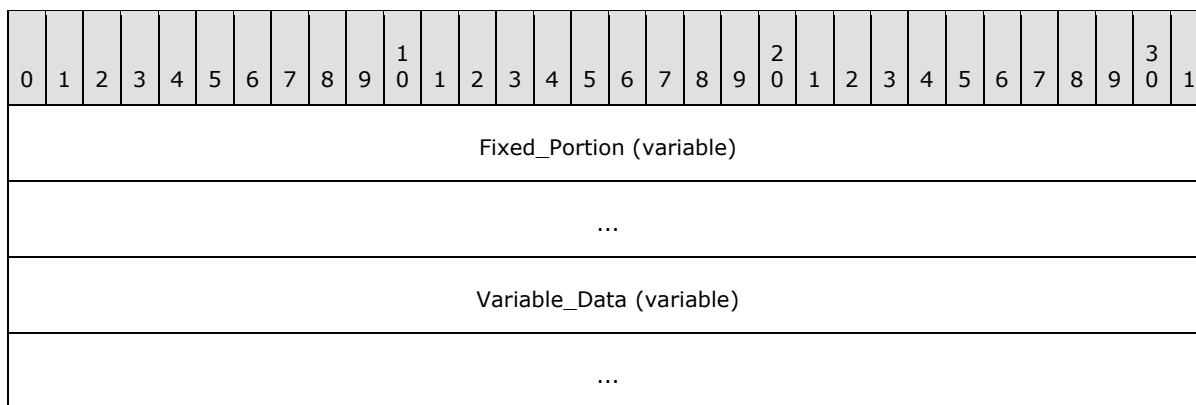
Variable_Data (variable): An array of zero or more optional, variable-size fields, which are defined as follows.



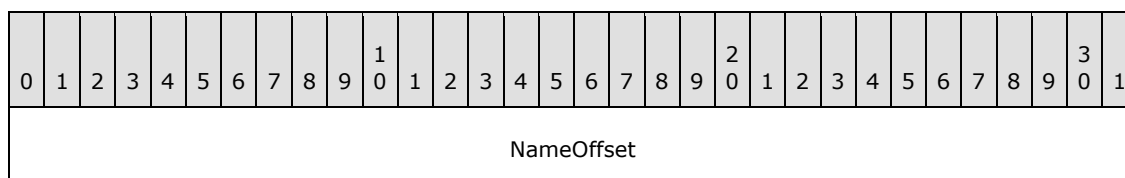
PathArray (variable): This member **MUST** be a string that **MUST** specify the path to the file to be printed. The location of this member is determined by the value of the **PathOffset** member.

2.2.2.4 DATATYPES_INFO_1

The DATATYPES_INFO_1 structure contains information about the datatype used to record a print job.

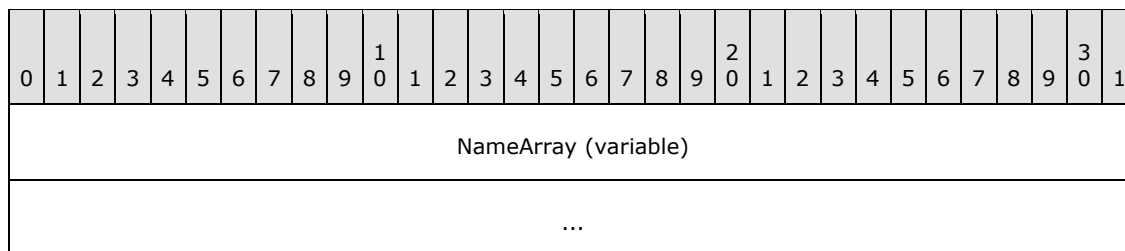


Fixed_Portion (variable): An array of one or more fixed-size fields, which are defined as follows.



NameOffset (4 bytes): This member is a 32-bit unsigned integer that MUST specify the number of bytes from the start of the structure to the **NameArray** member.

Variable_Data (variable): An array of zero or more optional, variable-size fields, which are defined as follows.

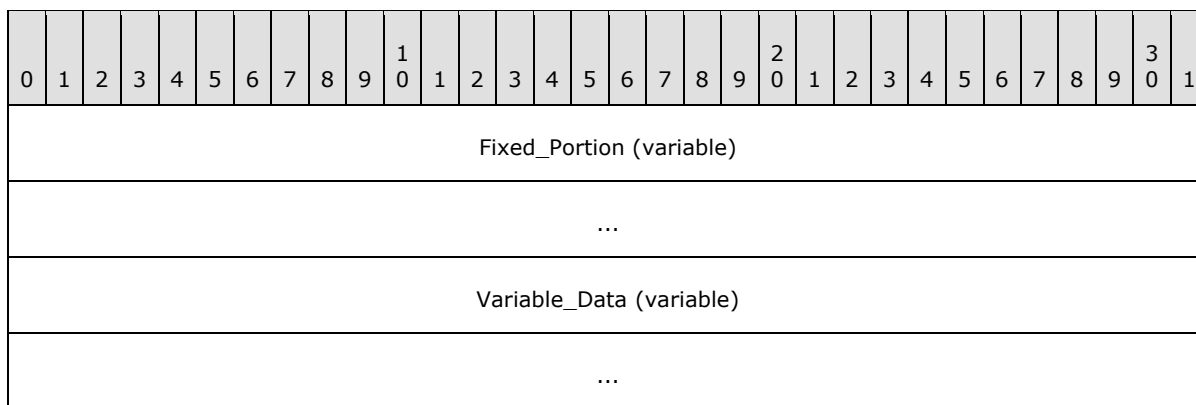


NameArray (variable): This member is a string that MUST specify the datatype used to record a print job. The location of this buffer is determined by the value of the **NameOffset** member. For rules governing datatype names, see section [2.2.4.15](#).

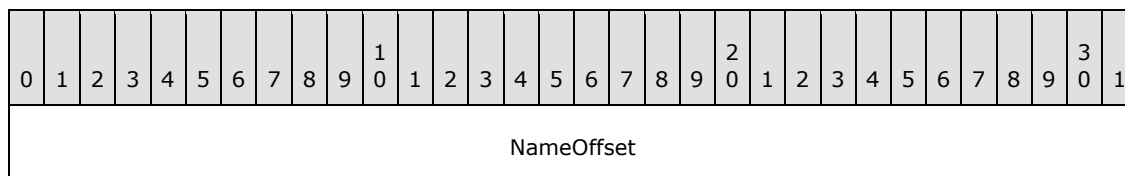
2.2.2.5 _DRIVER_INFO

2.2.2.5.1 _DRIVER_INFO_1

The **_DRIVER_INFO_1** structure specifies printer driver information. It is a custom-marshaled form of the [DRIVER_INFO_1 \(section 2.2.1.5.1\)](#) structure.

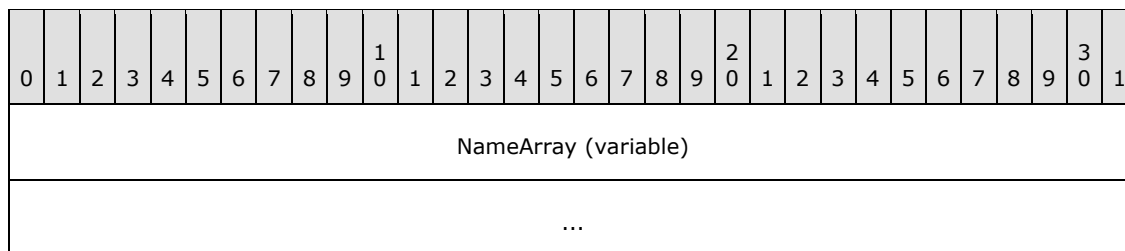


Fixed_Portion (variable): An array of one or more fixed-size fields, which are defined as follows.



NameOffset (4 bytes): This member is a 32-bit unsigned integer that **MUST** specify the number of bytes from the start of the structure to the **NameArray** member.

Variable_Data (variable): An array of zero or more optional, variable-size fields, which are defined as follows.



NameArray (variable): This is a string that **MUST** specify the datatype used to record a print job. The location of this buffer is determined by the value of the **NameOffset** member. For rules governing datatype names, see section [2.2.4.15](#).

2.2.2.5.2 _DRIVER_INFO_2

The **_DRIVER_INFO_2** structure specifies printer driver information. It is a custom-marshaled form of the [DRIVER_INFO_2 \(section 2.2.1.5.2\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cVersion																															
NameOffset																															
EnvironmentOffset																															
DriverPathOffset																															
DataFileOffset																															
ConfigFileOffset																															

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
ConfigFileArray (variable)																															
...																															
DataFileArray (variable)																															
...																															
DriverPathArray (variable)																															
...																															
EnvironmentArray (variable)																															
...																															
NameArray (variable)																															
...																															

2.2.2.5.3 _DRIVER_INFO_3

The _DRIVER_INFO_3 structure specifies printer driver information. It is a custom-marshaled form of the [RPC_DRIVER_INFO_3 \(section 2.2.1.5.3\)](#) structure.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
cVersion																															
NameOffset																															
EnvironmentOffset																															
DriverPathOffset																															
DataFileOffset																															
ConfigFileOffset																															
HelpFileOffset																															
DependentFilesOffset																															
MonitorNameOffset																															
DefaultDataTypeOffset																															

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
DefaultDataTypeArray (variable)																															
...																															
MonitorNameArray (variable)																															
...																															
DependentFilesArray (variable)																															
...																															
HelpFileArray (variable)																															

...
ConfigFileArray (variable)
...
DataFileArray (variable)
...
DriverPathArray (variable)
...
EnvironmentArray (variable)
...
NameArray (variable)
...

2.2.2.5.4 **_DRIVER_INFO_4**

The **_DRIVER_INFO_4** structure specifies printer driver information. It is a custom-marshaled form of the [RPC_DRIVER_INFO_4 \(section 2.2.1.5.4\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
cVersion																															
NameOffset																															
EnvironmentOffset																															
DriverPathOffset																															
DataFileOffset																															
ConfigFileOffset																															
HelpFileOffset																															
DependentFilesOffset																															
MonitorNameOffset																															
DefaultDataTypeOffset																															
szzPreviousNamesOffset																															

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
szzPreviousNamesArray (variable)																															
...																															
DefaultDataTypeArray (variable)																															
...																															
MonitorNameArray (variable)																															
...																															

DependentFilesArray (variable)
...
HelpFileArray (variable)
...
ConfigFileArray (variable)
...
DataFileArray (variable)
...
DriverPathArray (variable)
...
EnvironmentArray (variable)
...
NameArray (variable)
...

2.2.2.5.5 **_DRIVER_INFO_5**

The `_DRIVER_INFO_5` structure specifies printer driver information.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cVersion																															
NameOffset																															
EnvironmentOffset																															
DriverPathOffset																															
DataFileOffset																															
ConfigFileOffset																															
dwDriverAttributes																															
dwConfigVersion																															
dwDriverVersion																															

dwDriverAttributes (4 bytes): The value of this member MUST specify the printer driver attributes. The value of this member MUST be zero or a constant value from the following table:

Name/Value	Meaning
PRINTER_DRIVER_PACKAGE_AWARE 0x00000001	The driver is part of a driver package.

dwConfigVersion (4 bytes): The value of this member MUST specify the number of times the printer driver's configuration file has been upgraded (replaced with a newer binary) or downgraded (replaced with an older binary) since the system was restarted.

dwDriverVersion (4 bytes): The value of this member MUST specify the number of times the printer driver's executable file has been upgraded (replaced with a newer binary) or downgraded (replaced with an older binary) since the system was restarted.

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
ConfigFileArray (variable)																															
...																															
DataFileArray (variable)																															
...																															
DriverPathArray (variable)																															
...																															
EnvironmentArray (variable)																															
...																															
NameArray (variable)																															
...																															

Member definitions for all members not defined in this section are identical to members in [DRIVER_INFO_4 \(section 2.2.2.5.4\)](#).

2.2.2.5.6 _DRIVER_INFO_6

The DRIVER_INFO_6 structure specifies printer driver information. It is a custom-marshaled form of the [RPC_DRIVER_INFO_6 \(section 2.2.1.5.5\)](#) structure.

dwIDriverVersion
...
MfgNameOffset
OEMUrlOffset
HardwareIDOffset
ProviderOffset

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ProviderArray (variable)																															
...																															
HardwareIDArray (variable)																															
...																															
OEMUrlArray (variable)																															
...																															
MfgNameArray (variable)																															
...																															
szzPreviousNamesArray (variable)																															
...																															
DefaultDataTypeArray (variable)																															
...																															

MonitorNameArray (variable)
...
DependentFilesArray (variable)
...
HelpFileArray (variable)
...
ConfigFileArray (variable)
...
DataFileArray (variable)
...
DriverPathArray (variable)
...
EnvironmentArray (variable)
...
NameArray (variable)
...

2.2.2.5.7 **_DRIVER_INFO_7**

The `_DRIVER_INFO_7` structure specifies printer driver information.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cbSize																															
cVersion																															
szDriverNameOffset																															
szInfNameOffset																															
szInstallSourceRootOffset																															

cbSize (4 bytes): The value of this member MUST specify the size, in bytes, of the `_DRIVER_INFO_7` data structure.

szDriverNameOffset (4 bytes): This member is synonymous with the **NameOffset** member.

szInfNameOffset (4 bytes): This member is a 32-bit unsigned integer that MUST specify the number of bytes from the start of the structure to the start of the **szInfNameArray** member.

szInstallSourceRootOffset (4 bytes): This member is a 32-bit unsigned integer that MUST specify the number of bytes from the start of the structure to the start of the **szInstallSourceRootArray** member.

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
szInstallSourceRootArray (variable)																															
...																															
szInfNameArray (variable)																															
...																															
szDriverNameArray (variable)																															
...																															

szInstallSourceRootArray (variable): This member MUST be a string that specifies the path from which the driver MUST be installed. The location of this buffer is determined by the value of the **szInstallSourceRootOffset** member.

szInfNameArray (variable): This member MUST be a string that specifies the name of the driver's installation configuration file. The location of this buffer is determined by the value of the **szInfNameOffset** member. [<49>](#)

szDriverNameArray (variable): This member is synonymous with the **NameArray** member.

2.2.2.5.8 _DRIVER_INFO_8

The **_DRIVER_INFO_8** structure specifies printer driver information. It is a custom-marshaled form of the [RPC DRIVER_INFO_8 \(section 2.2.1.5.6\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
cVersion																															
NameOffset																															
EnvironmentOffset																															
DriverPathOffset																															
DataFileOffset																															
ConfigFileOffset																															
HelpFileOffset																															
DependentFilesOffset																															
MonitorNameOffset																															
DefaultDataTypeOffset																															
szzPreviousNamesOffset																															
ftDriverDate																															
...																															
dwlDriverVersion																															
...																															
MfgNameOffset																															
OEMUrlOffset																															
HardwareIDOffset																															
ProviderOffset																															

PrintProcessorOffset
VendorSetupOffset
szzColorProfilesOffset
InfPathOffset
dwPrinterDriverAttributes
szzCoreDependenciesOffset
ftMinInboxDriverVerDate
...
dwlMinInboxDriverVerVersion
...

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
InfPathArray (variable)																															
...																															
szzColorProfilesArray (variable)																															
...																															
VendorSetupArray (variable)																															
...																															
PrintProcessorArray (variable)																															
...																															

ProviderArray (variable)
...
HardwareIDArray (variable)
...
OEMUrlArray (variable)
...
MfgNameArray (variable)
...
szzPreviousNamesArray (variable)
...
DefaultDataTypeArray (variable)
...
MonitorNameArray (variable)
...
DependentFilesArray (variable)
...
HelpFileArray (variable)
...
ConfigFileArray (variable)
...
DataFileArray (variable)

...
DriverPathArray (variable)
...
EnvironmentArray (variable)
...
NameArray (variable)
...
szzCoreDependenciesArray (variable)
...

2.2.2.5.9 _DRIVER_INFO_101

The DRIVER_INFO_101 structure specifies printer driver information.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cVersion																															
NameOffset																															

EnvironmentOffset
FileInfoOffset
dwFileCount
MonitorNameOffset
DefaultDataTypeOffset
szzPreviousNamesOffset
ftDriverDate
...
dwIDriverVersion
...
MfgNameOffset
OEMUrlOffset
HardwareIDOffset
ProviderOffset

FileInfoOffset (4 bytes): This member is a 32-bit unsigned integer that MUST specify the number of bytes from the start of the structure to the start of the **FileInfoArray** member.

dwFileCount (4 bytes): This member is a 32-bit unsigned integer that MUST specify the number of [DRIVER_FILE_INFO](#) structures in the **FileTypeInfoArray** member.

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ProviderArray (variable)																															

...
HardwareIDArray (variable)
...
OEMUrlArray (variable)
...
MfgNameArray (variable)
...
szzPreviousNamesArray (variable)
...
DefaultDataTypeArray (variable)
...
MonitorNameArray (variable)
...
FileInfoArray (variable)
...
EnvironmentArray (variable)
...
NameArray (variable)
...

FileInfoArray (variable): This member MUST be an array of DRIVER_FILE_INFO structures. The number of elements in the array MUST be the same as the value of the **dwFileCount** member.

Note: Member definitions for all members not defined in this section are identical to members in [DRIVER_INFO_8 \(section 2.2.2.5.8\)](#).

2.2.2.5.10 _DRIVER_FILE_INFO

The _DRIVER_FILE_INFO structure specifies information about a file belonging to a printer driver.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FileNameOffset																															
FileType																															
FileVersion																															

FileNameOffset (4 bytes): This member is a 32-bit unsigned integer that MUST specify the number of bytes from the start of the structure to the start of the **FileNameArray** member.

FileType (4 bytes): This member is a 32-bit integer that MUST specify the file type using one of the constant values from the following table.

Value	Meaning
0	The file is a rendering driver module executable.
1	The file is a configuration module executable.
2	The file is a driver data file.
3	The file is a driver help file.
4	The file is a dependent file with a type other than the above file types.

FileVersion (4 bytes): This field specifies the implementation-specific version of the file.

Variable_Data (variable): An array of zero or more optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FileNameArray (variable)																															
...																															

FileNameArray (variable): This member MUST be a string that MUST specify the name of the file. The location of this buffer is determined by the value of the **FileNameOffset** member.

2.2.2.6 _FORM_INFO

2.2.2.6.1 _FORM_INFO_1

The _FORM_INFO_1 structure specifies printer media information. It is a custom-marshaled form of the [FORM_INFO_1 \(section 2.2.1.6.1\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Flags																															
NameOffset																															
Size																															
...																															
ImageableArea																															
...																															
...																															
...																															

Variable_Data (variable): An array of zero or more optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
NameArray (variable)																															
...																															

All members not defined in this section are specified in sections [2.2.1.3.2](#) and [2.2.2.4](#).

2.2.2.6.2 _FORM_INFO_2

The _FORM_INFO_2 structure specifies printer media information. It is a custom-marshaled form of the [RPC FORM_INFO_2 \(section 2.2.1.6.2\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Flags																															
NameOffset																															
Size																															
...																															
ImageableArea																															
...																															
...																															
...																															
KeywordOffset																															
StringType																															
MuiDIIOffset																															
dwResourceID																															
DisplayNameOffset																															
wLangID																unused															

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
NameArray (variable)																															
...																															
KeywordArray (variable)																															
...																															
MuiDllArray (variable)																															
...																															
DisplayNameArray (variable)																															
...																															

2.2.2.7 _JOB_INFO

2.2.2.7.1 _JOB_INFO_1

The _JOB_INFO_1 structure specifies print job information. It is a custom-marshaled form of the [JOB_INFO_1 \(section 2.2.1.7.1\)](#) structure.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
JobId																															
PrinterNameOffset																															
MachineNameOffset																															
UserNameOffset																															
DocumentOffset																															
DatatypeOffset																															
StatusOffset																															
Status																															
Priority																															
Position																															
TotalPages																															
PagesPrinted																															
Submitted																															
...																															
...																															
...																															

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
StatusArray (variable)																															
...																															
DatatypeArray (variable)																															
...																															
DocumentArray (variable)																															
...																															
UserNameArray (variable)																															
...																															
MachineNameArray (variable)																															
...																															
PrinterNameArray (variable)																															
...																															

Fields that are not defined in this section are specified in section [2.2.1.3.3](#).

2.2.2.7.2 _JOB_INFO_2

The _JOB_INFO_2 structure specifies print job information. It is a custom-marshaled form of the [JOB_INFO_2 \(section 2.2.1.7.2\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
JobId																															
PrinterNameOffset																															
MachineNameOffset																															
UserNameOffset																															
DocumentOffset																															
NotifyNameOffset																															
DatatypeOffset																															
PrintProcessorOffset																															
ParametersOffset																															
DriverNameOffset																															
DevModeOffset																															
StatusOffset																															
SecurityDescriptorOffset																															

Status
Priority
Position
StartTime
UntilTime
TotalPages
Size
Submitted
...
...
...
Time
PagesPrinted

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
StatusArray (variable)																															
...																															
DevModeArray (variable)																															
...																															
DriverNameArray (variable)																															

...
ParametersArray (variable)
...
PrintProcessorArray (variable)
...
DatatypeArray (variable)
...
NotifyNameArray (variable)
...
DocumentArray (variable)
...
UserNameArray (variable)
...
MachineNameArray (variable)
...
PrinterNameArray (variable)
...

Fields that are not defined in this section are specified in section [2.2.1.3.3](#).

2.2.2.7.3 _JOB_INFO_3

The _JOB_INFO_3 structure specifies information about the order of print jobs, and it is used to alter the order of print jobs. It is a custom-marshaled form of the [JOB_INFO_3 \(section 2.2.1.7.3\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
JobId																															
NextJobId																															
Reserved																															

Fields that are not defined in this section are specified in sections [2.2.1.7.3](#), [2.2.1.3](#), and [2.2.2.3](#).

2.2.2.7.4 _JOB_INFO_4

The _JOB_INFO_4 structure specifies print job information. It is a custom-marshaled form of the [JOB_INFO_4 \(section 2.2.1.7.4\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
JobId																															

PrinterNameOffset
MachineNameOffset
UserNameOffset
DocumentOffset
NotifyNameOffset
DatatypeOffset
PrintProcessorOffset
ParametersOffset
DriverNameOffset
DevModeOffset
StatusOffset
SecurityDescriptorOffset
Status
Priority
Position
StartTime
UntilTime
TotalPages
Size
Submitted
...

...
...
Time
PagesPrinted
SizeHigh

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
StatusArray (variable)																															
...																															
DevModeArray (variable)																															
...																															
DriverNameArray (variable)																															
...																															
ParametersArray (variable)																															
...																															
PrintProcessorArray (variable)																															
...																															
DatatypeArray (variable)																															
...																															
NotifyNameArray (variable)																															

...
DocumentArray (variable)
...
UserNameArray (variable)
...
MachineNameArray (variable)
...
PrinterNameArray (variable)
...

Fields that are not defined in this section are specified in section [2.2.1.3.3](#).

2.2.2.8 **_MONITOR_INFO**

2.2.2.8.1 **_MONITOR_INFO_1**

The _MONITOR_INFO_1 structure identifies an installed port monitor. It is a custom-marshaled form of the [MONITOR_INFO_1 \(section 2.2.1.8.1\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8	9	30	1
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
NameOffset																															

Variable_Data (variable): An array of zero or more optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
NameArray (variable)																															
...																															

2.2.2.8.2 _MONITOR_INFO_2

The _MONITOR_INFO_2 structure is used to identify a port monitor. It is a custom-marshaled form of the [MONITOR_INFO_2 \(section 2.2.1.8.2\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
NameOffset																															
EnvironmentOffset																															
DLLNameOffset																															

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
DLLNameArray (variable)																															
...																															
EnvironmentArray (variable)																															
...																															
NameArray (variable)																															
...																															

2.2.2.9 _PORT_INFO

2.2.2.9.1 _PORT_INFO_1

The _PORT_INFO_1 structure specifies information about a printer port. It is a custom-marshaled form of the [PORT_INFO_1 \(section 2.2.1.9.1\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
NameOffset																															

Variable_Data (variable): An array of zero or more optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
NameArray (variable)																															
...																															

2.2.2.9.2 _PORT_INFO_2

The _PORT_INFO_2 structure specifies information about a printer port. It is a custom-marshaled form of the [PORT_INFO_2 \(section 2.2.1.9.2\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PortNameOffset																															
MonitorNameOffset																															
DescriptionOffset																															
fPortType																															
Reserved																															

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
DescriptionArray (variable)																															
...																															
MonitorNameArray (variable)																															
...																															
PortNameArray (variable)																															
...																															

2.2.2.10 _PRINTER_INFO

2.2.2.10.1 _PRINTER_INFO_STRESS

The _PRINTER_INFO_STRESS structure specifies printer diagnostic information. It is a custom-marshaled form of the [PRINTER_INFO_STRESS \(section 2.2.1.10.1\)](#) structure. This form of the _PRINTER_INFO_STRESS structure corresponds to an information **Level** value of zero.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
PrinterNameOffset																															
ServerNameOffset																															

cJobs
cTotalJobs
cTotalBytes
stUpTime
...
...
...
MaxcRef
cTotalPagesPrinted
dwGetVersion
fFreeBuild
cSpooling
cMaxSpooling
cRef
cErrorOutOfPaper
cErrorNotReady
cJobError
dwNumberOfProcessors
dwProcessorType
dwHighPartTotalBytes
cChangeID

dwLastError	
Status	
cEnumerateNetworkPrinters	
cAddNetPrinters	
wProcessorArchitecture	wProcessorLevel
cRefIC	
dwReserved2	
dwReserved3	

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
PrinterNameArray (variable)																															
...																															
ServerNameArray (variable)																															
...																															

2.2.2.10.2 _PRINTER_INFO_1

The _PRINTER_INFO_1 structure specifies printer information. It is a custom-marshaled form of the [PRINTER_INFO_1 \(section 2.2.1.10.2\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Flags																															
DescriptionOffset																															
NameOffset																															
CommentOffset																															

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
CommentArray (variable)																															
...																															
NameArray (variable)																															
...																															
DescriptionArray (variable)																															
...																															

2.2.2.10.3 _PRINTER_INFO_2

The _PRINTER_INFO_2 structure specifies printer information. It is a custom-marshaled form of the [PRINTER_INFO_2 \(section 2.2.1.10.3\)](#) structure.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
ServerNameOffset																															
PrinterNameOffset																															
ShareNameOffset																															

PortNameOffset
DriverNameOffset
CommentOffset
LocationOffset
DevModeOffset
SepFileOffset
PrintProcessorOffset
DatatypeOffset
ParametersOffset
SecurityDescriptorOffset
Attributes
Priority
DefaultPriority
StartTime
UntilTime
Status
cJobs
AveragePPM

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

...
ShareNameArray (variable)
...
PrinterNameArray (variable)
...
ServerNameArray (variable)
...

2.2.2.10.4 _PRINTER_INFO_3

The _PRINTER_INFO_3 structure specifies printer information. It is a custom-marshaled form of the [PRINTER_INFO_3 \(section 2.2.1.10.4\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SecurityDescriptorOffset																															

Variable_Data (variable): An array of zero or more optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
SecurityDescriptorArray (variable)																															
...																															

2.2.2.10.5 _PRINTER_INFO_4

The _PRINTER_INFO_4 structure specifies printer information. It is a custom-marshaled form of the [PRINTER_INFO_4 \(section 2.2.1.10.5\)](#) structure.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
PrinterNameOffset																															
ServerNameOffset																															
Attributes																															

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
ServerNameArray (variable)																															
...																															
PrinterNameArray (variable)																															
...																															

2.2.2.10.6 _PRINTER_INFO_5

The _PRINTER_INFO_5 structure specifies printer information. It is a custom-marshaled form of the [PRINTER_INFO_5 \(section 2.2.1.10.6\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PrinterNameOffset																															
PortNameOffset																															
Attributes																															
DeviceNotSelectedTimeout																															
TransmissionRetryTimeout																															

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PortNameArray (variable)																															
...																															
PrinterNameArray (variable)																															
...																															

2.2.2.10.7 _PRINTER_INFO_6

The _PRINTER_INFO_6 structure specifies printer information. It is a custom-marshaled form of the [PRINTER_INFO_6 \(section 2.2.1.10.7\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															

Fixed_Portion (variable): An array of one or more fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
dwStatus																															

dwStatus (4 bytes): The value of this field MUST specify the new port status as specified in section [2.2.1.9.3](#).

2.2.2.10.8 _PRINTER_INFO_7

The _PRINTER_INFO_7 structure specifies printer information. It is a custom-marshaled form of the **PRINTER_INFO_7 (section 2.2.1.10.8)** structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
szObjectGUIDOffset																															
dwAction																															

Variable_Data (variable): An array of zero or more optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
szObjectGUIDArray (variable)																															
...																															

2.2.2.10.9 _PRINTER_INFO_8

The _PRINTER_INFO_8 structure specifies printer information. It is a custom-marshaled form of the [PRINTER_INFO_8 \(section 2.2.1.10.9\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DevModeOffset																															

Variable_Data (variable): An array of zero or more optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DevModeArray (variable)																															
...																															

2.2.2.10.10 _PRINTER_INFO_9

The _PRINTER_INFO_9 structure specifies printer information. It is a custom-marshaled form of the [PRINTER_INFO_9 \(section 2.2.1.10.10\)](#) structure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more fixed-size fields, which are defined as follows.

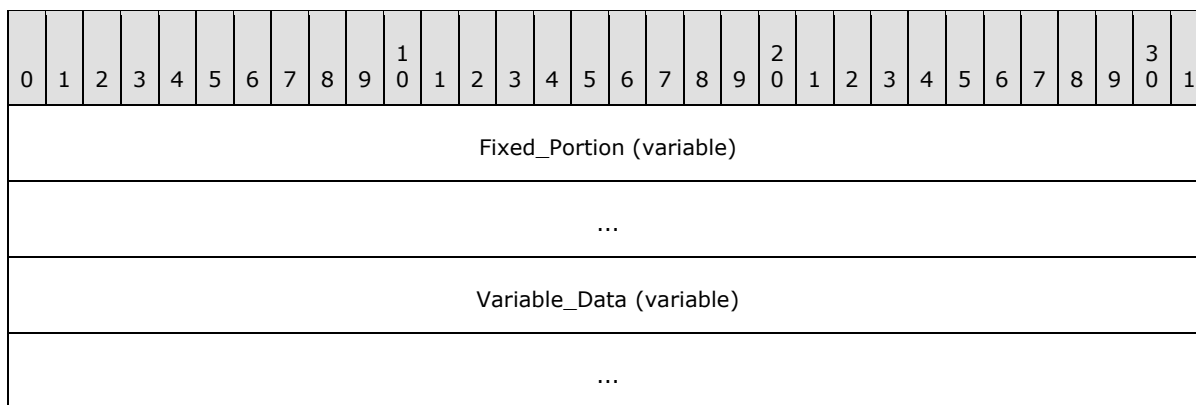
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DevModeOffset																															

Variable_Data (variable): An array of zero or more optional, variable-size fields, which are defined as follows.

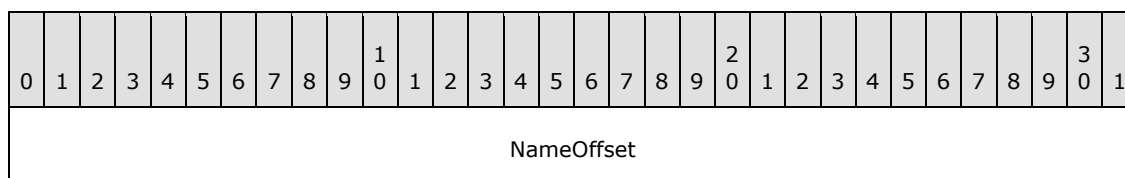
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DevModeArray (variable)																															
...																															

2.2.2.11 PRINTPROCESSOR_INFO_1

The PRINTPROCESSOR_INFO_1 structure specifies printer information.

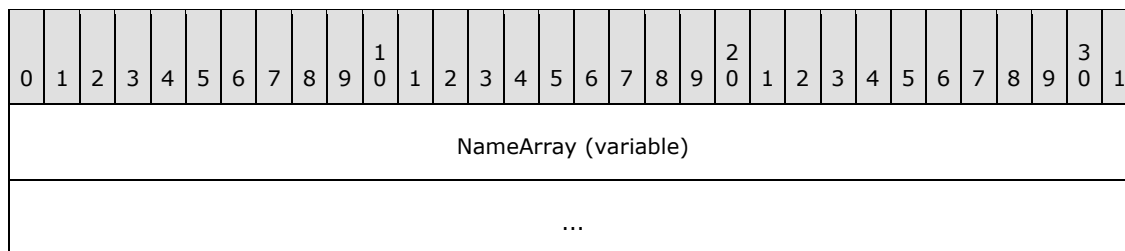


Fixed_Portion (variable): An array of one or more fixed-size fields, which are defined as follows.



NameOffset (4 bytes): This member is a 32-bit unsigned integer that MUST specify the number of bytes from the start of the structure to the start of the **NameArray** member.

Variable_Data (variable): An array of zero or more optional, variable-size fields, which are defined as follows.



NameArray (variable): This member MUST contain a string that specifies the print processor name. The location of this buffer is determined by the value of the **NameOffset** member. For rules governing print processor names, see section [2.2.4.8](#).

2.2.2.12 PRINTER_ENUM_VALUES

The PRINTER_ENUM_VALUES structure specifies the value name, type, and data for a printer configuration value.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Fixed_Portion (variable)																															
...																															
Variable_Data (variable)																															
...																															

Fixed_Portion (variable): An array of one or more groups of fixed-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ValueNameOffset																															
cbValueName																															
dwType																															
DataOffset																															
cbData																															

ValueNameOffset (4 bytes): This member is a 32-bit unsigned integer that MUST specify the number of bytes from the start of the structure to the start of the **ValueNameArray** member.

cbValueName (4 bytes): The value of this member MUST specify the size of the **ValueNameArray** in bytes.

dwType (4 bytes): The value of this member MUST specify the data type of the data in the **DataArray** member. For a list of the possible type codes, see section [2.2.3.9](#). For rules governing **registry** type values, see section [2.2.4.9](#).

DataOffset (4 bytes): This member is a 32-bit unsigned integer that MUST specify the number of bytes from the start of the structure to the start of the **Data** member.

cbData (4 bytes): The value of this member MUST specify the number of bytes retrieved in the **DataArray** buffer.

Variable_Data (variable): An array of zero or more groups of optional, variable-size fields, which are defined as follows.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
ValueNameArray (variable)																															
...																															
DataArray (variable)																															
...																															

ValueNameArray (variable): This member MUST contain a string that specifies the value name. The location of this buffer is determined by the value of the **ValueNameOffset** member. For rules governing value names, see section [2.2.4.16](#).

DataArray (variable): This member MUST contain the data for the retrieved value.

2.2.3 Constants

2.2.3.1 Status and Attribute Values

Printer Status	Description
PRINTER_STATUS_BUSY 0x00000200	The printer is busy.
PRINTER_STATUS_DOOR_OPEN 0x00400000	The printer door is open.
PRINTER_STATUS_ERROR 0x00000002	The printer is in an error state.
PRINTER_STATUS_INITIALIZING 0x00008000	The printer is initializing.
PRINTER_STATUS_IO_ACTIVE 0x00000100	The printer is in an active input/output state.
PRINTER_STATUS_MANUAL_FEED 0x00000020	The printer is in a manual feed state.
PRINTER_STATUS_NOT_AVAILABLE 0x00001000	The printer is not available for printing.
PRINTER_STATUS_NO_TONER 0x00040000	The printer is out of toner.
PRINTER_STATUS_OFFLINE 0x00000080	The printer is offline.
PRINTER_STATUS_OUTPUT_BIN_FULL 0x00000800	The printer's output bin is full.
PRINTER_STATUS_OUT_OF_MEMORY	The printer has run out of memory.

Printer Status	Description
0x00200000	
PRINTER_STATUS_PAGE_PUNT 0x00080000	The printer cannot print the current page.
PRINTER_STATUS_PAPER_JAM 0x00000008	Paper is stuck in the printer.
PRINTER_STATUS_PAPER_OUT 0x00000010	The printer is out of paper.
PRINTER_STATUS_PAPER_PROBLEM 0x00000040	The printer has an unspecified paper problem.
PRINTER_STATUS_PAUSED 0x00000001	The printer is paused.
PRINTER_STATUS_PENDING_DELETION 0x00000004	The printer is being deleted as a result of a client's call to <code>RpcDeletePrinter</code> . No new jobs can be submitted on existing printer objects for that printer.
PRINTER_STATUS_POWER_SAVE 0x01000000	The printer is in power-save mode.
PRINTER_STATUS_PRINTING 0x00000400	The printer is printing.
PRINTER_STATUS_PROCESSING 0x00004000	The printer is processing a print job.
PRINTER_STATUS_SERVER_OFFLINE 0x02000000	The printer is offline.
PRINTER_STATUS_SERVER_UNKNOWN 0x00800000	The printer status is unknown.
PRINTER_STATUS_TONER_LOW 0x00020000	The printer is low on toner.
PRINTER_STATUS_USER_INTERVENTION 0x00100000	The printer has an error that requires the user to do something.
PRINTER_STATUS_WAITING 0x00002000	The printer is waiting.
PRINTER_STATUS_WARMING_UP 0x00010000	The printer is warming up.

Printer Attribute	Description
PRINTER_ATTRIBUTE_DEFAULT 0x00000004	Indicates the printer is the default printer in the system.
PRINTER_ATTRIBUTE_DIRECT 0x00000002	Job is sent directly to the printer (it is not spooled).

Printer Attribute	Description
PRINTER_ATTRIBUTE_DO_COMPLETE_FIRST 0x00000200	If set and printer is set for print-while-spooling, any jobs that have completed spooling are scheduled to print before jobs that have not completed spooling.
PRINTER_ATTRIBUTE_ENABLE_BIDI 0x00000800	Indicates whether bidirectional communications are enabled for the printer.
PRINTER_ATTRIBUTE_ENABLE_DEVQ 0x00000080	Setting this flag causes mismatched documents to be held in the queue.
PRINTER_ATTRIBUTE_FAX 0x00004000	If set, printer is a fax printer.
PRINTER_ATTRIBUTE_KEEPPRINTEDJOBS 0x00000100	If set, jobs are kept after they are printed. If cleared, jobs are deleted.
PRINTER_ATTRIBUTE_LOCAL 0x00000040	Printer is a local printer.
PRINTER_ATTRIBUTE_NETWORK 0x00000010	Printer is a network printer connection.
PRINTER_ATTRIBUTE_PUBLISHED 0x00002000	Indicates whether the printer is published in the directory service.
PRINTER_ATTRIBUTE_QUEUED 0x00000001	If set, the printer spools and starts printing after the last page is spooled. If cleared, and PRINTER_ATTRIBUTE_DIRECT is not set, the printer spools and prints while spooling.
PRINTER_ATTRIBUTE_RAW_ONLY 0x00001000	Indicates that only RAW data type print jobs MUST be spooled.
PRINTER_ATTRIBUTE_SHARED 0x00000008	Printer is shared.
PRINTER_ATTRIBUTE_TS 0x00008000	Printer is a redirected terminal server printer.
PRINTER_ATTRIBUTE_WORK_OFFLINE 0x00000400	Indicates whether the printer is currently connected. If the printer is not currently connected, print jobs will continue to spool.

Job Status	Description
JOB_STATUS_BLOCKED_DEVQ 0x00000200	The printer driver cannot print the job.
JOB_STATUS_COMPLETE 0x00001000	The job has been delivered to the printer.
JOB_STATUS_DELETED 0x00000100	Job has been deleted.
JOB_STATUS_DELETING	Job is being deleted.

Job Status	Description
0x00000004	
JOB_STATUS_ERROR 0x00000002	An error is associated with the job.
JOB_STATUS_OFFLINE 0x00000020	Printer is offline.
JOB_STATUS_PAPEROUT 0x00000040	Printer is out of paper.
JOB_STATUS_PAUSED 0x00000001	Job is paused.
JOB_STATUS_PRINTED 0x00000080	Job has printed.
JOB_STATUS_PRINTING 0x00000010	Job is printing.
JOB_STATUS_RESTART 0x00000800	Job has been restarted.
JOB_STATUS_SPOOLING 0x00000008	Job is spooling.
JOB_STATUS_USER_INTERVENTION 0x00000400	Printer has an error that requires the user to do something.

2.2.3.2 Flags Values

Constant/value	Description
PRINTER_ENUM_CONNECTIONS 0x00000004	Enumerate printer connections previously added through RpcAddPerMachineConnection .
PRINTER_ENUM_CONTAINER 0x00008000	Indicates that the printer object is capable of containing enumerable objects.
PRINTER_ENUM_DEFAULT 0x00000001	Use the default printer object enumeration. <50>
PRINTER_ENUM_EXPAND 0x00004000	Indicates that the printer object contains further enumerable child objects.
PRINTER_ENUM_FAVORITE 0x00000004	Enumerate printer connections previously added through RpcAddPerMachineConnection .
PRINTER_ENUM_LOCAL 0x00000002	Enumerate local printer objects.
PRINTER_ENUM_HIDE 0x01000000	Indicates that an application SHOULD NOT display the printer object.
PRINTER_ENUM_NAME 0x00000008	Enumerate printers on the server, domain, or a specific print provider.

Constant/value	Description
PRINTER_ENUM_NETWORK 0x00000040	Enumerate network printers that are in the same domain as the print server.
PRINTER_ENUM_REMOTE 0x00000010	Enumerate network printers and other print servers that are in the same domain as the print server.
PRINTER_ENUM_SHARED 0x00000020	Only enumerate printers with the shared attribute set. This flag MUST be combined with one of the other flags.
PRINTER_ENUM_ICON1 0x00010000	Indicates that, where appropriate, an application SHOULD treat an object as a top-level network name, such as "Windows Network".<51> A GUI application MAY choose to display an icon of choice for this type of object.
PRINTER_ENUM_ICON2 0x00020000	Indicates that, where appropriate, an application SHOULD treat an object as a network domain name. A GUI application MAY<52> choose to display an icon of choice for this type of object.
PRINTER_ENUM_ICON3 0x00040000	Indicates that, where appropriate, an application SHOULD treat an object as a print server. A GUI application MAY<53> choose to display an icon of choice for this type of object.
PRINTER_ENUM_ICON4 0x00080000	Reserved.<54>
PRINTER_ENUM_ICON5 0x00100000	Reserved.<55>
PRINTER_ENUM_ICON6 0x00200000	Reserved.<56>
PRINTER_ENUM_ICON7 0x00400000	Reserved.<57>
PRINTER_ENUM_ICON8 0x00800000	Indicates that, where appropriate, an application SHOULD treat an object as a print server. A GUI application MAY<58> choose to display an icon of choice for this type of object.

2.2.3.3 Printer Change Values

Constant/value	Description
PRINTER_CHANGE_ADD_PRINTER 0x00000001	A printer object was added.
PRINTER_CHANGE_SET_PRINTER 0x00000002	Printer object properties were configured.
PRINTER_CHANGE_DELETE_PRINTER 0x00000004	A printer object was deleted.
PRINTER_CHANGE_FAILED_CONNECTION_PRINTER 0x00000008	A connection to a printer object failed.
PRINTER_CHANGE_PRINTER 0x000000FF	A printer object changed in some way.

Constant/value	Description
PRINTER_CHANGE_ADD_JOB 0x00000100	A print job was added.
PRINTER_CHANGE_SET_JOB 0x00000200	Print job properties were configured.
PRINTER_CHANGE_DELETE_JOB 0x00000400	A print job was deleted.
PRINTER_CHANGE_WRITE_JOB 0x00000800	A print job was written.
PRINTER_CHANGE_JOB 0x0000FF00	A print job changed in some way.
PRINTER_CHANGE_ADD_FORM 0x00010000	A form was added.
PRINTER_CHANGE_SET_FORM 0x00020000	Form properties were configured.
PRINTER_CHANGE_DELETE_FORM 0x00040000	A form was deleted.
PRINTER_CHANGE_FORM 0x00070000	A form was changed in some way.
PRINTER_CHANGE_ADD_PORT 0x00100000	A port was added.
PRINTER_CHANGE_CONFIGURE_PORT 0x00200000	Port properties were configured.
PRINTER_CHANGE_DELETE_PORT 0x00400000	A port was deleted.
PRINTER_CHANGE_PORT 0x00700000	A port was changed in some way.
PRINTER_CHANGE_ADD_PRINT_PROCESSOR 0x01000000	A print processor was added.
PRINTER_CHANGE_DELETE_PRINT_PROCESSOR 0x04000000	A print processor was deleted.
PRINTER_CHANGE_PRINT_PROCESSOR 0x07000000	The properties for a print processor were updated.
PRINTER_CHANGE_ADD_PRINTER_DRIVER 0x10000000	A printer driver was added.
PRINTER_CHANGE_SET_PRINTER_DRIVER 0x20000000	A printer driver was specified.
PRINTER_CHANGE_DELETE_PRINTER_DRIVER 0x40000000	A printer driver was deleted.
PRINTER_CHANGE_PRINTER_DRIVER	A printer driver was changed in some way.

Constant/value	Description
0x70000000	
PRINTER_CHANGE_TIMEOUT 0x80000000	Returned by RpcWaitForPrinterChange if the implementation-specific timeout has expired.
PRINTER_CHANGE_ALL 0x7777FFFF	A change was made to one or more printer-related objects, including print job, form, port, processor, or printer driver, or to the printer object itself.

For more information about the rules governing printer change values, see section [2.2.4.10](#).

2.2.3.4 Access Values

Name/Value	Description
JOB_ACCESS_ADMINISTER 0x00000010	Authorization to cancel, pause, resume, or restart the job.
JOB_ACCESS_READ 0x00000020	Read rights for the spool file.
JOB_EXECUTE 0x00020018	Access rights for jobs combining STANDARD_RIGHTS_EXECUTE , JOB_ACCESS_ADMINISTER , and PRINTER_ACCESS_USE .
JOB_READ 0x00020030	Access rights for jobs combining STANDARD_RIGHTS_REQUIRED , JOB_ACCESS_READ , and JOB_ACCESS_ADMINISTER .
JOB_WRITE 0x00020018	Access rights for jobs combining STANDARD_RIGHTS_WRITE , JOB_ACCESS_ADMINISTER , and PRINTER_ACCESS_USE .
PRINTER_ACCESS_ADMINISTER 0x00000004	Access rights for printers to perform administrative tasks.
PRINTER_ACCESS_USE 0x00000008	Access rights for printers to perform basic printing operations.
PRINTER_ALL_ACCESS 0x000F000C	Access rights for printers to perform all administrative tasks and basic printing operations except SYNCHRONIZE . Combines STANDARD_RIGHTS_REQUIRED , PRINTER_ACCESS_ADMINISTER , and PRINTER_ACCESS_USE .
PRINTER_EXECUTE 0x00020008	Access rights for printers combining STANDARD_RIGHTS_EXECUTE and PRINTER_ACCESS_USE .
PRINTER_READ 0x00020008	Access rights for printers combining STANDARD_RIGHTS_READ and PRINTER_ACCESS_USE .
PRINTER_WRITE 0x00020008	Access rights for printers combining STANDARD_RIGHTS_WRITE and PRINTER_ACCESS_USE .
SERVER_ACCESS_ADMINISTER 0x00000001	Access rights to administer print servers.
SERVER_ACCESS_ENUMERATE 0x00000002	Access rights to enumerate print servers.
SERVER_ALL_ACCESS	Access rights for print servers to perform all administrative tasks and

Name/Value	Description
0x000F0003	basic printing operations except SYNCHRONIZE . Combines STANDARD_RIGHTS_REQUIRED , SERVER_ACCESS_ADMINISTER , and SERVER_ACCESS_ENUMERATE .
SERVER_EXECUTE 0x00020002	Access rights for print servers combining STANDARD_RIGHTS_EXECUTE and SERVER_ACCESS_ENUMERATE .
SERVER_READ 0x00020002	Access rights for print servers combining STANDARD_RIGHTS_READ and SERVER_ACCESS_ENUMERATE .
SERVER_WRITE 0x00020003	Access rights for print servers combining STANDARD_RIGHTS_WRITE , SERVER_ACCESS_ADMINISTER , and SERVER_ACCESS_ENUMERATE .
SPECIFIC_RIGHTS_ALL 0x0000FFFF	All specific rights.
STANDARD_RIGHTS_ALL 0x001F0000	Combines DELETE , READ_CONTROL , WRITE_DAC , WRITE_OWNER , and SYNCHRONIZE access.
STANDARD_RIGHTS_EXECUTE 0x00020000	Standard rights, set to READ_CONTROL .
STANDARD_RIGHTS_READ 0x00020000	Standard READ_CONTROL rights. The right to read the information in the object's security descriptor, not including the information in the System Access Control List (SACL).
STANDARD_RIGHTS_REQUIRED 0x000F0000	Standard rights, combines DELETE , READ_CONTROL , WRITE_DAC , and WRITE_OWNER access.
STANDARD_RIGHTS_WRITE 0x00020000	Standard write rights, set to READ_CONTROL .
SYNCHRONIZE 0x00100000	The right to use the object for synchronization.
WRITE_DAC 0x00040000	The right to modify the Discretionary Access Control List (DACL) in the object's security descriptor.
WRITE_OWNER 0x00080000	The right to change the owner in the object's security descriptor.

2.2.3.5 Printer Notification Values

For detailed descriptions of the **Reserved** and **Data** values in the right column of the following table, refer to the [RPC_V2_NOTIFY_INFO_DATA \(section 2.2.1.13.4\)](#) structure. For all other values, refer to [PRINTER_INFO \(section 2.2.2.10\)](#) structures.

Name/Value	Description
PRINTER_NOTIFY_FIELD_ATTRIBUTES 0x000D	Printer attributes changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new Attributes value.

Name/Value	Description
PRINTER_NOTIFY_FIELD_AVERAGE_PPM 0x0015	Average pages per minute for the printer changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new AveragePPM value.
PRINTER_NOTIFY_FIELD_BYTES_PRINTED 0x0019	Number of bytes that have been printed changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new cTotalBytes value.
PRINTER_NOTIFY_FIELD_CJOBS 0x0014	Number of print jobs that have been queued for the printer changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new cJobs value.
PRINTER_NOTIFY_FIELD_COMMENT 0x0005	Printer comment changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new comment value (see pComment).
PRINTER_NOTIFY_FIELD_DATATYPE 0x000B	Printer default data type changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new comment value (see pDatatype).
PRINTER_NOTIFY_FIELD_DEFAULT_PRIORITY 0x000F	Default priority for the printer changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new DefaultPriority value.
PRINTER_NOTIFY_FIELD_DEVMODE 0x0007	Default DEVMODE for the printer changed. Reserved MUST contain TABLE_DEVMODE, and Data.DevMode MUST contain the new DEVMODE value (see pDevMode).
PRINTER_NOTIFY_FIELD_DRIVER_NAME 0x0004	Printer driver for the printer changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new driver name value (see pDriverName).
PRINTER_NOTIFY_FIELD_LOCATION 0x0006	Printer location changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new location description value (see pLocation).
PRINTER_NOTIFY_FIELD_OBJECT_GUID 0x001A	Printer object GUID changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new printer GUID value (see pszObjectGUID).
PRINTER_NOTIFY_FIELD_PAGES_PRINTED 0x0017	Number of pages that have been printed for the printer changed. Reserved MUST contain TABLE_DWORD, and

Name/Value	Description
	Data.dwData[0] MUST contain the new PagesPrinted value.
PRINTER_NOTIFY_FIELD_PARAMETERS 0x000A	Default print processor parameters for the printer changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new print processor parameters value (see pParameters).
PRINTER_NOTIFY_FIELD_PORT_NAME 0x0003	Default port name for the printer changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new port name value (see pPortName).
PRINTER_NOTIFY_FIELD_PRINTER_NAME 0x0001	Printer name changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new printer name value (see pPrinterName).
PRINTER_NOTIFY_FIELD_PRINT_PROCESSOR 0x0009	Print processor associated with the printer changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new print processor name value (see pPrintProcessor).
PRINTER_NOTIFY_FIELD_PRIORITY 0x000E	Current priority for the printer changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new Priority value.
PRINTER_NOTIFY_FIELD_SECURITY_DESCRIPTOR 0x000C	Security descriptor for the printer changed. Reserved MUST contain TABLE_SECURITY_DESCRIPTOR, and Data.SecurityDescriptor MUST contain the new security descriptor value (see pSecurityDescriptor).
PRINTER_NOTIFY_FIELD_SEPFILE 0x0008	The separator page for the printer changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new separator page value (see pSepFile).
PRINTER_NOTIFY_FIELD_SERVER_NAME 0x0000	The server name for the printer changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new server name value (see pServerName).
PRINTER_NOTIFY_FIELD_SHARE_NAME 0x0002	Printer share name changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new share name value (see pShareName).
PRINTER_NOTIFY_FIELD_START_TIME 0x0010	Earliest start time for the printer changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new StartTime value.

Name/Value	Description
PRINTER_NOTIFY_FIELD_STATUS 0x0012	Status for the printer changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new Status value.
PRINTER_NOTIFY_FIELD_TOTAL_BYTES 0x0018	Total bytes printed on the printer changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new cTotalBytes value.
PRINTER_NOTIFY_FIELD_TOTAL_PAGES 0x0016	Number of total pages that have been printed on the printer changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new cTotalPagesPrinted value.
PRINTER_NOTIFY_FIELD_UNTIL_TIME 0x0011	Latest print time for the printer changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new UntilTime value.

2.2.3.6 Job Notification Values

For detailed descriptions of the **Reserved** and **Data** values in the right column of the following table, refer to the [RPC_V2_NOTIFY_INFO_DATA \(section 2.2.1.13.4\)](#) structure. For all other values, refer to the [JOB_INFO \(section 2.2.2.7\)](#) structures.

Constant/value	Description
JOB_NOTIFY_FIELD_PRINTER_NAME 0x0000	Printer name for the job changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new printer name value (see pPrinterName).
JOB_NOTIFY_FIELD_MACHINE_NAME 0x0001	The server name for the job changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new server name value (see pMachineName).
JOB_NOTIFY_FIELD_PORT_NAME 0x0002	Port for the job changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain a string specifying the new port the job is printed on (see pPortName).
JOB_NOTIFY_FIELD_USER_NAME 0x0003	User name for the job changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new user name value (see pUserName).
JOB_NOTIFY_FIELD_NOTIFY_NAME 0x0004	Notify name for the job changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new notify name value (see pNotifyName).

Constant/value	Description
JOB_NOTIFY_FIELD_DATATYPE 0x0005	Default data type for the job changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new default data type value (see pDatatype).
JOB_NOTIFY_FIELD_PRINT_PROCESSOR 0x0006	Print processor associated with the job changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new print processor name value (see pPrintProcessor).
JOB_NOTIFY_FIELD_PARAMETERS 0x0007	Default print processor parameters for the job changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new print processor parameters value (see pParameters).
JOB_NOTIFY_FIELD_DRIVER_NAME 0x0008	Printer driver for the job changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new printer driver name value (see pDriverName).
JOB_NOTIFY_FIELD_DEVMODE 0x0009	Default DEVMODE for the job changed. Reserved MUST contain TABLE_DEVMODE, and Data.DevMode MUST contain the new DEVMODE value (see DEVMODE).
JOB_NOTIFY_FIELD_STATUS 0x000A	Status for the job changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new Status value.
JOB_NOTIFY_FIELD_STATUS_STRING 0x000B	The textual representation for the job status changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new status string value (see pStatus).
JOB_NOTIFY_FIELD_SECURITY_DESCRIPTOR 0x000C	Security descriptor for the job changed. Reserved MUST contain TABLE_SECURITY_DESCRIPTOR, and Data.SecurityDescriptor MUST contain the new security descriptor value (see pSecurityDescriptor).
JOB_NOTIFY_FIELD_DOCUMENT 0x000D	The document name for the job changed. Reserved MUST contain TABLE_STRING, and Data.String MUST contain the new document name value (see pDocument).
JOB_NOTIFY_FIELD_PRIORITY 0x000E	Current priority for the job changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new Priority value.
JOB_NOTIFY_FIELD_POSITION 0x000F	Position in the queue for the job changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new Position value.
JOB_NOTIFY_FIELD_SUBMITTED 0x0010	Submitted time for the job changed. Reserved MUST contain TABLE_TIME, and

Constant/value	Description
	Data.SystemTime MUST contain the new Submitted value.
JOB_NOTIFY_FIELD_START_TIME 0x0011	Earliest start time for the job changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new StartTime value.
JOB_NOTIFY_FIELD_UNTIL_TIME 0x0012	Latest print time for the job changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new UntilTime value.
JOB_NOTIFY_FIELD_TIME 0x0013	Total print time for the job changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new Time value.
JOB_NOTIFY_FIELD_TOTAL_PAGES 0x0014	Number of total pages of the job changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new cTotalPages value.
JOB_NOTIFY_FIELD_PAGES_PRINTED 0x0015	Number of pages that have been printed of the job changed. Reserved MUST contain TABLE_DWORD, and Data.dwData[0] MUST contain the new PagesPrinted value.
JOB_NOTIFY_FIELD_TOTAL_BYTES 0x0016	Total bytes of the job changed. Reserved MUST contain TABLE_DWORD, Data.dwData[0] MUST contain the new Size value, and Data.dwData[1] MUST contain the new SizeHigh value.
JOB_NOTIFY_FIELD_BYTES_PRINTED 0x0017	Total bytes printed of the job changed. Reserved MUST contain TABLE_DWORD, Data.dwData[0] MUST contain the 32 low-order bits of the number of bytes printed, and Data.dwData[1] MUST contain the 32 high-order bits of the number of bytes printed.

2.2.3.7 Change Notification Flag Values

The following bit flags MUST be set by the server to indicate the state of the notification structure, [RPC_V2_NOTIFY_INFO](#):

Name/Value	Description
PRINTER_NOTIFY_INFO_DISCARDED 0x00000001	MUST be set by the server if an overflow or error occurred, and notifications have been lost. Server MUST NOT send additional notifications until the client has made an RpcRouterRefreshPrinterChangeNotification call.
All other bits SHOULD be zero.	

A bitwise OR combination of zero or more of the following bit flags MUST be set by the client in the variable pointed to by the **pdwResult** parameter in [RpcRouterReplyPrinterEx](#) used to indicate how the client processed the state of the notification structure, **RPC_V2_NOTIFY_INFO**:

Name/Value	Description
PRINTER_NOTIFY_INFO_DISCARDNOTED 0x00010000	SHOULD be set by the client in <code>RpcRouterReplyPrinterEx</code> in the DWORD pointed to by pdwResult if it acknowledged receiving and processing the <code>PRINTER_NOTIFY_INFO_DISCARDED</code> notification.
PRINTER_NOTIFY_INFO_COLORMISMATCH 0x00080000	SHOULD be set by the client in <code>RpcRouterReplyPrinterEx</code> in the DWORD pointed to by pdwResult if the passed in dwColor value does not match the dwColor value the client previously passed to the server in RpcRouterRefreshPrinterChangeNotification .
All other bits SHOULD be zero.	

A bitwise OR combination of zero or more of the following bit flags MUST be set by the client in the notification options structure, [RPC_V2_NOTIFY_OPTIONS](#):

Name/Value	Description
PRINTER_NOTIFY_OPTIONS_REFRESH 0x00000001	The client MUST set this flag to request refreshed data from the server for all monitored members.
All other bits SHOULD be zero.	

2.2.3.8 Server Handle Key Values

Details concerning Server Handle Key Values can be found in [Appendix B: Windows Behavior.<59>](#)

2.2.3.9 Registry Type Values

Registry Type Name/Value	Description
REG_NONE 0x00000000	No value type is defined.
REG_SZ 0x00000001	A string.
REG_EXPAND_SZ 0x00000002	A string that MAY contain unexpanded references to environment variables, for example, "%PATH%".
REG_BINARY 0x00000003	Binary data in any form.
REG_DWORD 0x00000004	A 32-bit number.
REG_DWORD_LITTLE_ENDIAN 0x00000004	A 32-bit number in little-endian format; equivalent to REG_DWORD.

Registry Type Name/Value	Description
REG_DWORD_BIG_ENDIAN 0x00000005	A 32-bit number in big-endian format.
REG_LINK 0x00000006	Symbolic link to a registry key.
REG_MULTI_SZ 0x00000007	A multisz.
REG_RESOURCE_LIST 0x00000008	Device-driver resource list.
REG_QWORD 0x0000000B	A 64-bit number.
REG_QWORD_LITTLE_ENDIAN 0x0000000B	A 64-bit number in little-endian format; equivalent to REG_QWORD.

2.2.3.10 BIDI_TYPE Enumeration

The **BIDI_TYPE Enumeration** enumeration specifies the type of data transferred in a bidirectional operation.

```
typedef enum
{
    BIDI_NULL = 0x00000000,
    BIDI_INT = 0x00000001,
    BIDI_FLOAT = 0x00000002,
    BIDI_BOOL = 0x00000003,
    BIDI_STRING = 0x00000004,
    BIDI_TEXT = 0x00000005,
    BIDI_ENUM = 0x00000006,
    BIDI_BLOB = 0x00000007
} BIDI_TYPE;
```

BIDI_NULL: No bidirectional data.

BIDI_INT: Bidirectional data is an integer.

BIDI_FLOAT: Bidirectional data is a floating-point number.

BIDI_BOOL: Bidirectional data is a Boolean value.

BIDI_STRING: Bidirectional data is a string.

BIDI_TEXT: Bidirectional data is text data.

BIDI_ENUM: Bidirectional data is an enumeration.

BIDI_BLOB: Bidirectional data is a data **BLOB**.

2.2.4 Rules for Members

The following sections specify rules for the common string and flag members that are passed as parameters, or are parts of structures that are passed as parameters, to methods in this protocol.

2.2.4.1 Server Names

A **server name** MUST specify the name of a print server. The value MUST be a string containing a **DNS**, **NetBIOS**, **IPv4**, **IPv6** or **UNC** name, or a string with length zero. An empty server name string identifies the same server that the **RPC server endpoint** is bound to. Server names MUST follow the **SERVER_NAME** pattern.

The **SERVER_NAME** pattern is defined as follows:

```
SERVER_NAME = "\\\" host "\" | ""
```

The **SERVER_NAME_NE** pattern is used where a non-empty server name is required and is defined as follows:

```
SERVER_NAME_NE = "\\\" host
```

For further information on **DNS** names, see [\[RFC819\]](#) section 2. For **NetBIOS** names, details are specified in [\[RFC1001\]](#) section 14. Details about basic notational conventions are specified in [\[RFC2616\]](#) section 2. The definition of "host" is specified in [\[RFC3986\]](#) section 3.2.2. [<60>](#)

2.2.4.2 Printer Names

A **printer name** on a print server MUST be unique and MUST NOT be a zero length string. The client or server MAY impose minimum or maximum length restrictions on this parameter.

In [RpcOpenPrinter](#) and [RpcOpenPrinterEx](#), all instances of printer names MUST follow either the **PRINTER_NAME** or **PRINTER_NAME_EX** pattern. In any other context, printer names MUST follow the **PRINTER_NAME** pattern.

The **PRINTER_NAME** pattern is defined as follows:

```
UNICODE_NOCOMMA_NOBACKSLASH = <Any UTF-16LE character except ",",  
and "\">
```

```
UNICODE_NOBACKSLASH = <Any UTF-16LE character, except "\">
```

```
PRINTER_NAME = (SERVER_NAME LOCAL_PRINTER_NAME) |  
(WEB_PRINT_SERVER "/" "printers" "/" LOCAL_PRINTER_NAME "/"  
".printer")
```

```
WEB_PRINT_SERVER = "http: " "/" host [":" port]
```

```
LOCAL_PRINTER_NAME = 1#UNICODE_NOCOMMA_NOBACKSLASH
```

where:

- **SERVER_NAME** is defined in section [2.2.4.1](#).

- **WEB_PRINT_SERVER** specifies the address of the Web print server.
- **LOCAL_PRINTER_NAME** is a string specifying the local printer name or share name of the printer. Printer names MUST NOT contain the characters ',' and '\'.

Basic notational conventions are specified in [\[RFC2616\]](#) section 2, and the definitions of "host" and "port" are specified in [\[RFC3986\]](#) section 3.2.2.

The **PRINTER_NAME_EX** pattern extends the **PRINTER_NAME** pattern and is defined as follows:

```
PRINTER_NAME_EX = PRINTER_NAME_EX1 | PRINTER_NAME_EX2 |
    PRINTER_NAME_EX3 | PRINTER_NAME_EX4 | PRINTER_NAME_EX5

PRINTER_NAME_EX1 = SERVER_NAME LOCAL_PRINTER_NAME "," # " "
    "Job " # " " JOB_IDENTIFIER

JOB_IDENTIFIER = MUST be between 1 and 2,147,483,648, inclusive.

PRINTER_NAME_EX2 = SERVER_NAME LOCAL_PORT_NAME "," # " " "Port"

LOCAL_PORT_NAME = 1#UNICODE_NOCOMMA

PRINTER_NAME_EX3 = SERVER_NAME_NE "," # " " "XcvPort "
    XCV_PORT_NAME

XCV_PORT_NAME = 1#UNICODE_NOCOMMA

PRINTER_NAME_EX4 = SERVER_NAME_NE "," # " " "XcvMonitor "
    XCV_MONITOR_NAME

XCV_MONITOR_NAME = 1#UNICODE_NOBACKSLASH

PRINTER_NAME_EX5 = SERVER_NAME_NE
```

where:

- **PRINTER_NAME_EX1** specifies the print job identified by **JOB_IDENTIFIER** on the printer specified with **PrinterName**.

When the **PRINTER_NAME_EX1** form is used as the name parameter with **RpcOpenPrinter** or **RpcOpenPrinterEx**, a job object will be returned that can be used with [RpcReadPrinter](#) and [RpcWritePrinter](#) to read and write job content.

- **JOB_IDENTIFIER** specifies a server-wide unique decimal identifier for the print job.
- **PRINTER_NAME_EX2** specifies the name of the port to be opened.

When the **PRINTER_NAME_EX2** form is used as the name parameter with **RpcOpenPrinter** or **RpcOpenPrinterEx**, a port object MUST be returned that can be used with [RpcStartDocPrinter](#) and **RpcWritePrinter** to print directly to a port without intermediate spooling.

- **LOCAL_PORT_NAME** is a port name, as specified in section [2.2.4.13](#).
- **PRINTER_NAME_EX3** specifies the **XcvPort** to be opened.

When the **PRINTER_NAME_EX3** form is used as the name parameter with **RpcOpenPrinter** or **RpcOpenPrinterEx**, a port object will be returned that can be used with [RpcXcvData](#) to communicate directly with an **XcvPort**.

- **XCV_PORT_NAME** is a port name, as specified in section [2.2.4.13](#).
- **PRINTER_NAME_EX4** specifies the **Xcv** port monitor to be opened.

When this form is used as the name parameter with **RpcOpenPrinter** or **RpcOpenPrinterEx**, a port object that can be used with **RpcXcvData** to communicate directly with an Xcv port monitor **MUST** be returned.

- **XCV_MONITOR_NAME** is a monitor name, as specified in section [2.2.4.12](#).
- **PRINTER_NAME_EX5** specifies the print server to be opened. [<61>](#)

2.2.4.3 Access Values

Access values specify the access rights that a caller is requesting.

2.2.4.4 Job Control Values

Job control values specify actions such as pause and cancel (see [RpcSetJob \(section 3.1.4.3.1\)](#) for a list of job control values and their meanings). A job control value **MUST NOT** be zero if a job control action is required.

2.2.4.5 Environment Names

An **environment name** **MUST** specify a string that **MUST** contain the name of the operating system environment. The string **MUST NOT** be empty and **MUST** permit white space. [<62>](#)

2.2.4.6 Driver Names

A **driver name** **MUST** specify a string that **MUST** contain the name of a printer driver. The string **MUST NOT** be empty. [<63>](#)

2.2.4.7 Path Names

A **path name** **MUST** specify a string that **MUST** contain the file name, or full path and file name, for the identified file. The string **MUST NOT** be empty.

A path name **MAY** contain any number of subdirectories, as indicated on the pattern below by **DIRECTORY**.

If the value indicates a network addressable file, the value **MUST** be a string containing a DNS, NetBIOS, IPv4, IPv6, or UNC name. The string **SHOULD** be of the form "\\ServerName\ShareName" and **MUST** identify a unique shared folder on the machine. White space **MUST** be permitted.

For further information on DNS names, see [\[RFC819\]](#) section 2. NetBIOS names are specified in [\[RFC1001\]](#) section 14.

If the path name indicates a local file, the interpretation is implementation dependent. [<64>](#)

2.2.4.8 Print Processor Names

A **print processor name** MUST specify a string that MUST contain the name of a print processor. The string MUST NOT be empty and MUST uniquely identify a print processor for a given operating system environment. [<65>](#)

2.2.4.9 Registry Type Values

A **registry type value** MUST specify the type of a data value in the registry.

2.2.4.10 Printer Change Values

The **Printer Change Value** MUST specify the changes on the print server. The flag MUST be zero or more of the values defined in section [2.2.3.3](#).

2.2.4.11 Form Names

A **form name** MUST specify a string that MUST contain the name of a printer form. The string MUST NOT be empty and it MUST uniquely identify a printer form on the system. [<66>](#)

2.2.4.12 Monitor Names

A **monitor name** MUST specify a string that MUST contain the name of a print monitor. The string MUST NOT be empty and it MUST uniquely identify a port monitor on the system. [<67>](#)

2.2.4.13 Port Names

A **port name** MUST specify a string that MUST contain the name of a printer port. The string MUST NOT be empty. The constraints for port names depend on the port monitor that they belong to. These constraints are not protocol-specific and are defined by the implementers of port monitors. [<68>](#)

2.2.4.14 Print Provider Names

A **print provider name** MUST specify a string parameter containing the name of a print provider. The string MUST NOT be empty and MUST uniquely identify a print provider on the system. [<69>](#)

2.2.4.15 Datatype Names

A **datatype name** MUST specify a string that MUST contain the name of a datatype. The string MUST NOT be empty. It MUST uniquely identify a print data type supported by a print processor. [<70>](#)

2.2.4.16 Value Names

A **value name** MUST specify a string that MUST contain the name of a value. The string MUST NOT be empty. This value MUST be kept under a **printer key**. [<71>](#)

2.2.4.17 Key Names

A **key name** MUST specify a string that MUST contain the name of a **printer key**. It MUST permit a backslash ("\") as delimiter to permit a path with one or more subkeys. It MUST uniquely identify a path under the main registry key where printer configuration data is kept. [<72>](#)

2.2.4.18 User Names

A **user name** MUST specify a string that MUST contain the name of the user. The string MUST NOT be empty, it MUST specify a valid **principal**, and it MUST be formatted as either an **Active Directory (AD)** name or a name with a fully qualified domain name. For more information, see the "Object Names and Identities" section in [\[MSDN-ADOVRVW\]](#).

3 Protocol Details

The following sections specify details of the Print System Remote Protocol, including abstract data models, interface method syntax, and message processing rules.

3.1 Server Details

3.1.1 Abstract Data Model

This section describes a conceptual model of a possible data organization that an implementation might need to maintain in order to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

This protocol depends on an abstract data model that maintains printers and related objects. These objects represent physical output devices and are used in the protocol to communicate with the devices, print to them, and manage them. The print server **MUST** behave as if it hosted the following objects in the specified hierarchy.

- **List of Print Server Names:** The server name that is passed to the print server by the client **MAY** differ from the server name that the print server determined upon its own initialization; for example, when aliasing of server names via DNS or directory services takes place.

Other methods, such as [RpcAddPerMachineConnection](#), pass in an additional server name parameter that identifies a print server that is different from the one handling the API call. To correctly resolve server names, each print server **MUST** maintain a mapping between server names and server addresses. When composing a response to the client, the print server, in order to identify itself, **MUST** use the same name that the client passed as the parameter in the method call.

- **List of Printers:** Each printer represents a physical print device or a number of homogeneous physical devices installed on the print server. Each printer object **MUST** maintain the following data elements:
 - A name that uniquely identifies the printer.
 - A reference to a printer driver object for the printer.
 - A reference to a print processor object.
 - A reference to a port object.
 - A list of queued or printing print jobs.
 - Global [DEVMODE](#) settings.
 - Per-user [DEVMODE](#) settings.
- **List of Drivers:** Each printer driver represents the software component responsible for converting print content submitted by applications into device-specific commands. Each printer driver object **MUST** maintain the following data elements:
 - A name that uniquely identifies the printer driver.
 - A list of well-known modules (rendering module, configuration module, and data module).

Additionally, each printer driver object SHOULD maintain the following optional data elements:

- A list of dependent files.
- Information about the printer driver manufacturer, printer driver timestamp, and version.
- **List of Port Monitors:** A port monitor is a component that can send buffers of data to devices using supported protocols. Port monitors MAY manage extended communication with the device, such as collection status information from the device. Port monitors MAY expose zero or more ports.

Port monitor modules are implementation-specific for a given port type. A printer port identifies a device connected to the machine via an implementation-specific protocol understood by its parent port monitor.
- **List of Ports:** A port represents a connection to the actual print device. Ports are exposed and managed by port monitors.
- **List of Print Providers:** A print provider performs transparent routing of print system calls to the local spooler or to a remote spooler. Print providers are Windows-specific and not required by this protocol, but an implementation SHOULD enumerate at least one implementation-defined print provider name when [RpcEnumPrinters](#) is called.
- **List of Print Processors:** Print processors, provided by device manufacturers or generic suppliers, perform additional manipulation of print content before it is sent to the device.
- **List of Known Printers:** The server SHOULD maintain a list of known printers that includes printers not installed on the print server but installed on other print servers reachable on the network. The printers in this list MUST only be used when composing a response for **RpcEnumPrinters** with the appropriate flags set, as specified in **RpcEnumPrinters**. This list facilitates printer detection in networks without directory services.

The abstract model MUST be able to relate each printer to a single printer driver in addition to a single port. Additionally, each printer MUST be related to exactly one print processor.

For every object stored in the model, there MUST be an associated set of attributes, as described in **IDL Data Types** (section [2.2.1](#)) and **Custom-Marshaled Data Types** (section [2.2.2](#)).

At the implementer's option, objects from the abstract model MAY be stored persistently on disk or other suitable medium, or MAY be stored transiently; the protocol itself does not require any specific choice.<73> For information about default Windows registry configuration, see [Appendix B: Windows Behavior](#).<74>

However, it is the server's responsibility to ensure consistency among persistently stored objects and transient copies thereof. For locations of print system components in the Windows server file system, see Appendix B: Windows Behavior.<75> Additionally, it is the server's responsibility to manage any resources (memory, disk space, locks, physical ports, and so on) used for object representations, and properly manage the lifetime of these resources according to the objects' lifetime.

3.1.2 Timers

No protocol timer events are required on the client beyond timers for the underlying RPC protocol.

3.1.3 Initialization

The server SHOULD listen on the well-known endpoints defined for this RPC interface in section [2.1](#).

The server MUST perform initialization according to the following rules, when calling an RPC notification method on the client:

- Create an RPC binding handle to the server RPC endpoint (the client implements a server endpoint for this protocol in order to process notifications), or use an RPC context handle, as specified in [\[C706\]](#), section [2.3](#).
- Use RPC handles of the following types:
 - Context handles that are used across multiple calls to the client, for methods taking a [PRINTER_HANDLE](#).
 - Handles that are bound to a single call to the client, for name-based methods taking a [STRING_HANDLE](#). A **STRING_HANDLE_bind** method MUST be implemented by the server.
- When creating the **RPC binding handle** on the named pipe **\pipe\spoolss**, the server MUST specify an **ImpersonationLevel** of 2 (**Impersonation**), as specified in [\[MS-SMB2\]](#), section [2.2.13](#).

3.1.4 Message Processing Events and Sequencing Rules

The Print System Remote Protocol MUST indicate to the **RPC runtime**, as specified in [\[MS-RPCE\]](#) section , that it will do the following:

- Perform a strict NDR data consistency check at target level 6.0.
- Reject a NULL unique or full pointer with non-zero conformant value.
- Using the **strict_context_handle** attribute, reject the use of context handles created by the methods of a different RPC interface.

This section specifies the syntax and behavior for server-side methods defined in this protocol. These methods are grouped into functional categories, and their syntax and behavior are specified in sections corresponding to those categories, as shown in the following table:

Functional Category	Description	Section
Printer management and discovery	Methods used for discovering and obtaining access to supported printers.	3.1.4.2
Job management	Methods for discovering, defining, and scheduling print jobs.	3.1.4.3
Printer driver management	Methods for discovering and installing printer drivers.	3.1.4.4
Form management	Methods for discovering and configuring printer forms.	3.1.4.5
Printer port management	Methods for discovering and communicating with printer ports.	3.1.4.6
Port monitor management	Methods for discovering and installing port monitor modules.	3.1.4.7
Print processor management	Methods for discovering and manipulating print processor objects.	3.1.4.8
Document printing	Methods for printing documents, pages and data.	3.1.4.9
Notifications	Methods for obtaining notifications of printing events.	3.1.4.10

Functional Category	Description	Section
Monitor Modules	Methods specified by executable language monitors.	3.1.4.11

The following table lists all the methods of the Print System Remote Protocol in ascending **opnum** order.

Methods in RPC Opnum Order

Method	Description
RpcEnumPrinters	RpcEnumPrinters enumerates available printers, print servers, domains, or print providers. Opnum: 0
RpcOpenPrinter	RpcOpenPrinter retrieves a printer handle, port handle, job handle, or print server handle. Opnum: 1
RpcSetJob	RpcSetJob pauses, resumes, cancels, or restarts a print job. It also sets print job parameters, for example, the job priority and the document name. Opnum: 2
RpcGetJob	RpcGetJob retrieves information about a specified print job. Opnum: 3
RpcEnumJobs	RpcEnumJobs retrieves information about a specified set of print jobs for a specified printer. Opnum: 4
RpcAddPrinter	RpcAddPrinter adds a printer to the list of supported printers for a specified server. Opnum: 5
RpcDeletePrinter	RpcDeletePrinter deletes the specified printer object. Opnum: 6
RpcSetPrinter	RpcSetPrinter sets the data for a specified printer or sets the state of the specified printer by pausing or resuming printing, or clearing all print jobs. Opnum: 7
RpcGetPrinter	RpcGetPrinter retrieves information about a specified printer. Opnum: 8
RpcAddPrinterDriver	RpcAddPrinterDriver installs a printer driver on the print server and links the configuration, data, and printer driver files. Opnum: 9
RpcEnumPrinterDrivers	RpcEnumPrinterDrivers enumerates the printer

Method	Description
	drivers installed on a specified print server. Opnum: 10
RpcGetPrinterDriver	RpcGetPrinterDriver retrieves printer driver data for the specified printer. Opnum: 11
RpcGetPrinterDriverDirectory	RpcGetPrinterDriverDirectory retrieves the path of the printer driver directory. Opnum: 12
RpcDeletePrinterDriver	RpcDeletePrinterDriver removes the specified printer driver from the list of supported drivers for a server. Opnum: 13
RpcAddPrintProcessor	RpcAddPrintProcessor installs a print processor on the specified server and adds its name to an internal list of supported print processors. Opnum: 14
RpcEnumPrintProcessors	RpcEnumPrintProcessors enumerates the print processors installed on a specified server. Opnum: 15
RpcGetPrintProcessorDirectory	RpcGetPrintProcessorDirectory retrieves the path for the print processor on the specified server. Opnum: 16
RpcStartDocPrinter	RpcStartDocPrinter notifies the print spooler that a document is being spooled for printing. Opnum: 17
RpcStartPagePrinter	RpcStartPagePrinter notifies the spooler that a page is about to be printed on the specified printer. Opnum: 18
RpcWritePrinter	RpcWritePrinter sends data to the print spooler. Opnum: 19
RpcEndPagePrinter	RpcEndPagePrinter notifies the print spooler that the application is at the end of a page in a print job. Opnum: 20
RpcAbortPrinter	RpcAbortPrinter aborts the currently spooling print document. Opnum: 21
RpcReadPrinter	RpcReadPrinter retrieves data from the specified printer. Opnum: 22
RpcEndDocPrinter	RpcEndDocPrinter notifies the print spooler that the application is at the end of the current print job.

Method	Description
	Opnum: 23
RpcAddJob	RpcAddJob adds a print job to the list of jobs that the print spooler can schedule and retrieves the name of the file used to store the job. Opnum: 24
RpcScheduleJob	RpcScheduleJob schedules a print job created by calling RpcAddJob on the print spooler. Opnum: 25
RpcGetPrinterData	RpcGetPrinterData retrieves printer configuration data for a printer or print server. Opnum: 26
RpcSetPrinterData	RpcSetPrinterData sets the configuration data for a printer or print server. Opnum: 27
RpcWaitForPrinterChange	RpcWaitForPrinterChange (opnum 28) retrieves information about the most recent change notification associated with a printer or print server. Opnum: 28
RpcClosePrinter	RpcClosePrinter closes a handle to a printer object, server object, job object, or port object. Opnum: 29
RpcAddForm	RpcAddForm adds a form name to the list of supported forms. Opnum: 30
RpcDeleteForm	RpcDeleteForm removes a form name from the list of supported forms. Opnum: 31
RpcGetForm	RpcGetForm retrieves information about a specified form. Opnum: 32
RpcSetForm	RpcSetForm replaces the form information for the specified form. Opnum: 33
RpcEnumForms	RpcEnumForms enumerates the forms that the specified printer supports. Opnum: 34
RpcEnumPorts	RpcEnumPorts enumerates the ports that are available for printing on a specified server. Opnum: 35
RpcEnumMonitors	RpcEnumMonitors retrieves information about the port monitors installed on the specified server.

Method	Description
	Opnum: 36
Opnum37NotUsedOnWire	Reserved for local use. Opnum: 37
Opnum38NotUsedOnWire	Reserved for local use. Opnum: 38
RpcDeletePort	RpcDeletePort removes a port added by the RpcAddPortEx method. Opnum: 39
RpcCreatePrinterIC	RpcCreatePrinterIC is called by the graphics device interface (GDI) to create an information context for a printer. Opnum: 40
RpcPlayGdiScriptOnPrinterIC	RpcPlayGdiScriptOnPrinterIC returns identifying information for fonts available for printing to a printer object. Opnum: 41
RpcDeletePrinterIC	RpcDeletePrinterIC deletes a printer information context (IC). Opnum: 42
Opnum43NotUsedOnWire	Reserved for local use. Opnum: 43
Opnum44NotUsedOnWire	Reserved for local use. Opnum: 44
Opnum45NotUsedOnWire	Reserved for local use. Opnum: 45
RpcAddMonitor	RpcAddMonitor installs a local port monitor and links the configuration, data, and monitor files. Opnum: 46
RpcDeleteMonitor	RpcDeleteMonitor removes a port monitor. Opnum: 47
RpcDeletePrintProcessor	RpcDeletePrintProcessor removes a print processor. Opnum: 48
Opnum49NotUsedOnWire	Reserved for local use. Opnum: 49
Opnum50NotUsedOnWire	Reserved for local use. Opnum: 50
RpcEnumPrintProcessorDatatypes	RpcEnumPrintProcessorDatatypes enumerates the data types that a specified print processor

Method	Description
	supports. Opnum: 51
RpcResetPrinter	RpcResetPrinter resets the data type and device mode values to use for printing documents submitted by the RpcStartDocPrinter method (see section 3.1.4.9.1). Opnum: 52
RpcGetPrinterDriver2	RpcGetPrinterDriver2 retrieves printer driver data for the specified printer. Opnum: 53
Opnum54NotUsedOnWire	Reserved for local use. Opnum: 54
Opnum55NotUsedOnWire	Reserved for local use. Opnum: 55
RpcFindClosePrinterChangeNotification	RpcFindClosePrinterChangeNotification closes a change notification object created by calling the FindFirstPrinterChangeNotification function. The printer or print server associated with the change notification object will no longer be monitored by that object. Opnum: 56
Opnum57NotUsedOnWire	Reserved for local use. Opnum: 57
RpcReplyOpenPrinter	RpcReplyOpenPrinter establishes a context handle from the server to the client. The server uses the context handle to send notification data to the client machine. Opnum: 58
RpcRouterReplyPrinter	RpcRouterReplyPrinter handles the notification coming from a remote router (as opposed to one coming from a print provider). Opnum: 59
RpcReplyClosePrinter	RpcReplyClosePrinter closes the notification channel opened by the RpcReplyOpenPrinter (see section 3.2.4.1.1) method between the server and client. Opnum: 60
RpcAddPortEx	RpcAddPortEx adds a port name to the list of supported ports. Opnum: 61
RpcRemoteFindFirstPrinterChangeNotification	RpcRemoteFindFirstPrinterChangeNotification creates a remote change notification object that monitors changes to printer objects and sends change notifications to the client using the method

Method	Description
	RpcRouterReplyPrinter (see section 3.2.4.1.2). Opnum: 62
Opnum63NotUsedOnWire	Reserved for local use. Opnum: 63
Opnum64NotUsedOnWire	Reserved for local use. Opnum: 64
RpcRemoteFindFirstPrinterChangeNotificationEx	RpcRemoteFindFirstPrinterChangeNotificationEx creates a remote change notification object that monitors changes to printer objects and sends change notifications to the client using the method RpcRouterReplyPrinterEx (see section 3.2.4.1.4). Opnum: 65
RpcRouterReplyPrinterEx	RpcRouterReplyPrinterEx handles the notification coming from a remote router (as opposed to one coming from a print provider). Opnum: 66
RpcRouterRefreshPrinterChangeNotification	RpcRouterRefreshPrinterChangeNotification returns change notification information. Opnum: 67
Opnum68NotUsedOnWire	Reserved for local use. Opnum: 68
RpcOpenPrinterEx	RpcOpenPrinterEx retrieves a printer handle, port handle, job handle, or print server handle. Opnum: 69
RpcAddPrinterEx	RpcAddPrinterEx installs a printer on the print server. Opnum: 70
RpcSetPort	RpcSetPort sets the status associated with a printer port. Opnum: 71
RpcEnumPrinterData	RpcEnumPrinterData enumerates configuration data for a specified printer. Opnum: 72
RpcDeletePrinterData	RpcDeletePrinterData deletes specified configuration data for a printer. Opnum: 73
Opnum74NotUsedOnWire	Reserved for local use. Opnum: 74
Opnum75NotUsedOnWire	Reserved for local use. Opnum: 75

Method	Description
Opnum76NotUsedOnWire	Reserved for local use. Opnum: 76
RpcSetPrinterDataEx	RpcSetPrinterDataEx sets the configuration data for a printer or print server. This method is similar to RpcSetPrinterData (see section 3.1.4.2.8) but also allows the caller to specify the registry key under which to store the data. Opnum: 77
RpcGetPrinterDataEx	RpcGetPrinterDataEx retrieves configuration data for the specified printer or print server. This method extends RpcGetPrinterData (see section 3.1.4.2.7) and can retrieve values sorted under a specified key by RpcSetPrinterDataEx (see section 3.1.4.2.18). Opnum: 78
RpcEnumPrinterDataEx	RpcEnumPrinterDataEx enumerates all value names and data for a specified printer and key. This method extends RpcEnumPrinterData (see section 3.1.4.2.16) by retrieving several values in a single call. Opnum: 79
RpcEnumPrinterKey	RpcEnumPrinterKey enumerates the subkeys of a specified key for a specified printer. Printer data is stored in the registry. Opnum: 80
RpcDeletePrinterDataEx	RpcDeletePrinterDataEx deletes a specified value from a printer's configuration data, which consists of a set of named and typed values stored in a hierarchy of registry keys. Opnum: 81
RpcDeletePrinterKey	RpcDeletePrinterKey deletes a specified key and all of its subkeys for a specified printer. Opnum: 82
Opnum83NotUsedOnWire	Reserved for local use. Opnum: 83
RpcDeletePrinterDriverEx	RpcDeletePrinterDriverEx removes the specified printer driver from the list of supported drivers for a server and deletes the files associated with the printer driver. This method can also delete specific versions of the printer driver. Opnum: 84
RpcAddPerMachineConnection	RpcAddPerMachineConnection stores the print server name and the name of the binary executable used as a provider for a particular connection. Opnum: 85

Method	Description
RpcDeletePerMachineConnection	RpcDeletePerMachineConnection deletes information about a server and connection provider. Opnum: 86
RpcEnumPerMachineConnections	RpcEnumPerMachineConnections enumerates each of the connections and copies PRINTER_INFO_4 structures (see section 2.2.1.10.5) for all the per-machine connections to the buffer pPrinterEnum. Opnum: 87
RpcXcvData	RpcXcvData provides an extensible mechanism by which a client can control ports on the server and exchange port-specific commands and data with the server. See section 3.1.4.11 for details on language monitor methods. Opnum: 88
RpcAddPrinterDriverEx	RpcAddPrinterDriverEx installs a printer driver on the print server. This method performs a similar function as RpcAddPrinterDriver (see section 3.1.4.4.1) and additionally allows to specify options that permit strict upgrade, strict downgrade, copying of newer files only, and copying of all files (regardless of their time stamps). Opnum: 89
Opnum90NotUsedOnWire	Reserved for local use. Opnum: 90
Opnum91NotUsedOnWire	Reserved for local use. Opnum: 91
Opnum92NotUsedOnWire	Reserved for local use. Opnum: 92
Opnum93NotUsedOnWire	Reserved for local use. Opnum: 93
Opnum94NotUsedOnWire	Reserved for local use. Opnum: 94
Opnum95NotUsedOnWire	Reserved for local use. Opnum: 95
RpcFlushPrinter	RpcFlushPrinter is used by the printer driver to send a buffer of bytes to the port to cleanly abort a print job. It also allows delaying the I/O line to the printer. Opnum: 96
RpcSendRecvBidiData	RpcSendRecvBidiData sends and receives bidirectional data. This method is used to communicate with port monitors that support such data.

Method	Description
	Opnum: 97
Opnum98NotUsedOnWire	Reserved for local use. Opnum: 98
Opnum99NotUsedOnWire	Reserved for local use. Opnum: 99
Opnum100NotUsedOnWire	Reserved for local use. Opnum: 100
Opnum101NotUsedOnWire	Reserved for local use. Opnum: 101
Opnum102NotUsedOnWire	Reserved for local use. Opnum: 102
Opnum103NotUsedOnWire	Reserved for local use. Opnum: 103
Opnum104NotUsedOnWire	Reserved for local use. Opnum: 104
Opnum105NotUsedOnWire	Reserved for local use. Opnum: 105
Opnum106NotUsedOnWire	Reserved for local use. Opnum: 106
Opnum107NotUsedOnWire	Reserved for local use. Opnum: 107
Opnum108NotUsedOnWire	Reserved for local use. Opnum: 108
Opnum109NotUsedOnWire	Reserved for local use. Opnum: 109

In the table above, the term "Reserved for local use" means that the client MUST NOT send the opnum, and server behavior is undefined since it does not affect interoperability. <76>

All these methods are request/response RPC methods. They MUST return zero to indicate successful completion and nonzero values to indicate failure, except where specifically described.

Unless stated otherwise, if a method fails for any reason, returning a nonzero failure value, the server state as visible to the client through this or any other protocol MUST NOT change.

Two nonzero return codes have specific meanings in this protocol, ERROR_MORE_DATA and ERROR_INSUFFICIENT_BUFFER, as specified in [MS-ERREF]. When a method declaration in this specification has an output parameter that supplies a needed buffer size, one of the values in the table below MAY be returned from a call to that method to enable the caller to discover that size. Thus, there are circumstances in which a nonzero return value SHOULD NOT be treated as an error,

but, instead, the client SHOULD allocate a buffer with a larger size and retry the request. These cases are noted in the method definitions in this section.

Error Name/Code	Meaning
ERROR_INSUFFICIENT_BUFFER 0x0000007A	The data area passed to a system call is too small.
ERROR_MORE_DATA 0x000000EA	More data is available.

3.1.4.1 Commonly Used Parameters

This section describes parameters commonly used in method calls with consistent definitions. The type of each parameter is given by the method declaration in the relevant method sections.

Individual method sections specify only parameters whose definitions are different from, or that are not listed in, this section; however, they may impose additional restrictions on the values of parameters defined in this section.

3.1.4.1.1 Printer Name Parameters

pPrinterName: This parameter MUST be a non-null pointer to a string that MUST specify the name of the printer connection, printer object, server object, job object, or port object. The string that is referenced by this parameter MUST NOT be empty. For rules governing printer names, see section [2.2.4.2](#).

The individual method sections include the following parameter validation steps by reference:

- Verify that the string pointed to by **pPrinterName** is well-formed according to the rules governing printer names (see section [2.2.4.2](#)), and if that verification fails, return ERROR_INVALID_PRINTER_NAME.
- Verify that the string pointed to by **pPrinterName** parameter corresponds to a printer, server, job, or port name, and if that verification fails, return ERROR_INVALID_NAME.

3.1.4.1.2 Print Server Name Parameters

pName: This parameter is a non-null pointer to a string that specifies the name of the print server that the method operates on. This MUST be a **DNS**, **NetBIOS**, **IPv4**, **IPv6**, or Universal Naming Convention (UNC) name that **RPC** binds to, and it MUST uniquely identify a print server on the network.

For all methods taking a [STRING_HANDLE](#) custom binding handle parameter, the Print System Remote Protocol assumes that the bind routine provided by the client uses the name provided through the **pName** parameter to create the **RPC binding**, although that is not strictly necessary from an RPC perspective. Although it is possible to create an **RPC binding** to a different server than that identified by the **pName** parameter, the Print System Remote Protocol has not been designed and tested for that usage pattern. However, server implementations MAY choose to implement support for server names not identical to the server name used to create the **RPC binding**, and as a result effectively route the call to another server. [<77>](#)

Note Regardless of the above statement, server implementations MUST NOT assume that the server name passed via the **pName** parameter matches the name the server determined upon its own initialization; the server name passed in MAY differ from that name as a result of server name

aliasing, for example, by use of **DNS** names or directory services. The server **MUST** use the passed-in name to compose names for responses because the client is not aware that aliasing occurred.

RPC binding handles are specified in [\[C706\]](#) section [2.3](#). For rules governing server names, see section [2.2.4.1](#).

pServer: Synonymous with **pName**.

Name: Synonymous with **pName**.

The individual method sections include the following parameter validation steps by reference:

- Verify that the string pointed to by the **Name** parameter is well-formed according to the rules governing server names (see section [2.2.4.1](#)), and if that verification fails, return **ERROR_INVALID_NAME**.
- Verify that the string pointed to by the **Name** parameter corresponds to a server name, and if that verification fails, return **ERROR_INVALID_NAME**.

3.1.4.1.3 Datatype Name Parameters

pDatatype: This parameter **MUST** be one of the following:

- Null, to indicate that the default datatype for the printer **MUST** be used.
- A non-null pointer to a string that **MUST** specify the data type to be associated with the printer handle.

For rules governing datatype names, see section [2.2.4.15](#).

The individual method sections include the following parameter validation steps by reference:

- If **pDatatype** is not null, verify that the string that is referenced by the **pDatatype** parameter identifies one of the data types that the printer or print server supports, and if that verification fails, return **ERROR_INVALID_DATATYPE**.

3.1.4.1.4 Environment Name Parameters

pEnvironment: This parameter **MUST** either be null or a pointer to a string that **MUST** specify the environment name. For rules governing environment names and Windows behaviors, see section [2.2.4.5](#).

The individual method sections include the following parameter validation steps by reference:

- If **pEnvironment** is null, use the local environment of the print server.
- If **pEnvironment** is a non-null pointer, verify that the string that is referenced by the **pEnvironment** parameter identifies an environment name that is supported on this server, and if that verification fails, return **ERROR_INVALID_ENVIRONMENT**.

3.1.4.1.5 INFO Structures Query Parameters

Unless noted otherwise, methods that return one or more of the INFO structures (see sections [2.2.1.3](#) to [2.2.1.11](#)) use a common API pattern in which the caller **MUST** pass in the following parameters:

Level: If this parameter is present in the method signature, its value **MUST** be the desired Level of the INFO structures.

BUFFER: This parameter **MUST** be a buffer into which the server **MUST** copy the requested INFO structures. The term **BUFFER** is used here as a placeholder. Each method section defines which of its parameters contains the pointer to the buffer.

cbBuf: The value of this parameter **MUST** be the size, in bytes, of the buffer. The value of cbBuf **MAY** be larger than the required size for the requested INFO structures.

pcbNeeded: This parameter **MUST** be a non-null pointer to a variable into which the server **MUST** copy the actual number of bytes between the start of the buffer and the last byte written by the server into the buffer, both inclusive; or the required size of the buffer, in bytes, if the value of the **cbBuf** parameter is smaller than the actual size of the data to return to the caller.

For methods capable of returning more than one INFO structure, the caller **MUST** also pass in:

pcReturned: This parameter **MUST** be a non-null pointer to a variable into which the server **MUST** copy the actual number of INFO structures written to the buffer if **cbBuf** was of sufficient size to hold all INFO structures.

The individual method sections use the following documentation conventions:

- **BUFFER TYPE:** Type of the INFO structure(s) returned in the buffer, one of the following:
 - DATATYPES_INFO_1
 - _DRIVER_INFO
 - _FORM_INFO
 - _JOB_INFO
 - _MONITOR_INFO
 - _PORT_INFO
 - _PRINTER_INFO
 - PRINTPROCESSOR_INFO_1
- **Level:** Lists valid levels.

The individual method sections include the following parameter validation steps by reference:

- The server **MUST** verify that *Level* has a valid value, and if that verification fails, the server **MUST** return **ERROR_INVALID_LEVEL**.
- The server **MUST** verify that the value of **cbBuf** is not smaller than the number of bytes required to hold the INFO structures to be written to the buffer, and if that verification fails, the server **MUST** write the number of bytes required into the variable pointed to by **pcbNeeded** and return **ERROR_INSUFFICIENT_BUFFER**.
- If the value of **cbBuf** is not zero, the server **MUST** verify that the pointer to the buffer that was passed in is a non-null pointer, and if that verification fails, the server **MUST** return **ERROR_INVALID_USER_BUFFER**.

The individual method sections include the following processing and response steps by reference:

- The server MUST populate [BUFFER](#) with INFO structures of a type specified by **TYPE** that describe the objects enumerated according to method-specific enumeration steps.
- For methods capable of returning more than one INFO structure, the server MUST store the number of INFO structures that it writes to BUFFER in the variable pointed to by **pcReturned**.
- The server MUST return 0 for success, or a nonzero error code if the method was not successful.
- With the exception of diagnostic purposes, the server state as visible to the client through this or any other protocol MUST NOT change as a result of processing this call.

3.1.4.1.6 String Query Parameters

Unless noted otherwise, methods that return one or more string values use a common API pattern, in which the caller MUST pass in:

- **Level:** If this parameter is present in the method signature, the value of this parameter MUST be 0x00000001.
- **BUFFER:** This parameter MUST be a buffer into which the server MUST copy the requested string values. The term **BUFFER** is used here as a placeholder. Each method section defines which of its parameters is the pointer to the buffer. Methods capable of returning more than one string value MUST write the values to the buffer as a multisz.
- **cbBuf:** The value of this parameter MUST be the size, in bytes, of the buffer. The value of cbBuf MAY be larger than the required size for the requested string values.
- **pcbNeeded:** This parameter MUST be a non-null pointer to a variable into which the server MUST copy the actual number of bytes between the start of the buffer and the last byte written by the server into the buffer, both inclusive, or the required size of the buffer in bytes if the value of the **cbBuf** parameter is smaller than the actual size of the data to return to the caller.

For methods capable of returning more than one string value, the caller MUST also pass in:

- **pcReturned:** This parameter MUST be a non-null pointer to a variable into which the server MUST copy the actual number of string values written to the buffer if **cbBuf** was of sufficient size to hold all string values.

The individual method sections include the following parameter validation steps by reference:

- The server MUST verify that the value of **cbBuf** is not smaller than the number of bytes required to hold the string values to be written to the buffer. If that verification fails, the server MUST write the number of bytes required to the variable pointed to by the **pcbNeeded** parameter and return ERROR_INSUFFICIENT_BUFFER.
- If the value of the **cbBuf** parameter is not zero, the server MUST verify that a non-null pointer to the buffer was passed in. If that verification fails, the server MUST return ERROR_INVALID_USER_BUFFER.

The individual method sections include the following processing and response steps by reference:

- The server MUST populate [BUFFER](#) with string values enumerated according to method-specific enumeration steps. Multiple string values MUST be represented as multisz.
- For methods that are capable of returning more than one string value, the server MUST store the number of string values written to BUFFER into the variable pointed to by **pcReturned**.

- The server MUST return 0 for success or a nonzero error code if the method was not successful.
- With the exception of diagnostic purposes, the server state as visible to the client through this or any other protocol MUST NOT change as a result of processing this call.

3.1.4.1.7 Dynamically Typed Query Parameters

Unless noted otherwise, methods returning one or more dynamically typed values use a common API pattern, in which the caller MUST pass in:

- **BUFFER**: This parameter MUST be a buffer into which the server MUST copy the requested dynamically typed values. The term **BUFFER** is used here as a placeholder. Each method section defines which of its parameters is the pointer to the buffer.
- **pType**: This parameter MUST be null or it MUST be a non-null pointer to a variable that MUST receive a code that MUST indicate the type of data that is stored in the specified value. For a list of the possible type values, see section [2.2.3.9](#).
- **nSize** or **cbData**: This parameter MUST be the size, in bytes, of the buffer. The value of **nSize** or **cbData** MAY be larger than the required size for the requested dynamically typed values.
- **pcbNeeded** or **pcbData**: This parameter MUST be a non-null pointer to a variable into which the server MUST copy the actual number of bytes between the start of the buffer and the last byte written by the server into the buffer, both inclusive, or the required size of the buffer in bytes if the value of the **cbBuf** parameter is smaller than the actual size of the data to return to the caller.

For methods capable of returning more than one dynamically typed value, the caller MUST also pass in:

- **pcReturned**: This parameter MUST be a non-null pointer to a variable into which the server MUST copy the actual number of dynamically typed values that were written to the buffer, if **cbBuf** was of sufficient size to hold all dynamically typed values.

The individual method sections include the following parameter validation steps by reference:

- The server MUST verify that the value of **cbBuf** is not smaller than the number of bytes required to hold the dynamically typed values to be written to the buffer. If that verification fails, the server MUST write the number of bytes required into the variable that is pointed to by **pcbNeeded** and return `ERROR_MORE_DATA`.
- If the value of the **cbBuf** parameter is not zero, the server MUST verify that a non-null pointer to the buffer was passed in. If that verification fails, the server MUST return `ERROR_INVALID_USER_BUFFER`.

The individual method sections include the following processing and response steps by reference:

- The server MUST populate **BUFFER** with dynamically typed values enumerated according to method-specific enumeration steps.
- If **pType** is not a null pointer, the server MUST write the type of the data returned in **BUFFER** to the variable pointed to by **pType**.
- For methods capable of returning more than one dynamically typed value, the server MUST store the number of values that were written to **BUFFER** into the variable pointed to by **pcReturned**.
- The server MUST return 0 for success or a nonzero error code if the method was not successful.

With the exception of diagnostic purposes, the server state as visible to the client through this or any other protocol MUST NOT change as a result of processing this call.

3.1.4.1.8 PRINTER_ENUM_VALUES Structures Query Parameters

Unless noted otherwise, methods returning one or more [PRINTER_ENUM_VALUES](#) structures (see sections [2.2.1.3](#) to [2.2.1.11](#)) use a common API pattern, in which the caller MUST pass in:

- **BUFFER**: This parameter MUST be a buffer into which the server MUST copy the requested PRINTER_ENUM_VALUES structures. The term **BUFFER** is used here as a placeholder. Each method section defines which of its parameters points to the buffer.
- **cbEnumValues**: The value of this parameter MUST be the size, in bytes, of the buffer. The value of **cbEnumValues** MAY be larger than the required size for the requested PRINTER_ENUM_VALUES structures.
- **pcbEnumValues**: This parameter MUST be a non-null pointer to a variable into which the server MUST copy the actual size in bytes between the start of the buffer and the last byte written by the server into the buffer, both inclusive, or if **cbBuf** is smaller than the actual data size, MUST return the required size in bytes for the buffer.

For methods capable of returning more than one PRINTER_ENUM_VALUES structure, the caller MUST also pass in:

- **pnEnumValues**: This parameter MUST be a non-null pointer to a variable into which the server MUST copy the actual number of PRINTER_ENUM_VALUES structures written to the buffer if **cbEnumValues** was of sufficient size to hold all PRINTER_ENUM_VALUES structures.

The individual method sections include the following parameter validation steps by reference:

- The server MUST verify that the value of the **cbEnumValues** parameter is not smaller than the number of bytes required by the PRINTER_ENUM_VALUES structures to be written to the buffer. If that verification fails, the server MUST write the number of bytes required to the variable that is pointed to by **pcbEnumValues** and return ERROR_MORE_DATA.
- If the value of the **cbEnumValues** parameter is not zero, the server MUST verify that a non-null pointer to the buffer was passed in. If that verification fails, the server MUST return ERROR_INVALID_USER_BUFFER.

The individual method sections include the following processing and response steps by reference:

- The server MUST populate the [BUFFER](#) with PRINTER_ENUM_VALUES structures that describe the enumerated objects according to method-specific enumeration steps.
- For methods capable of returning more than one PRINTER_ENUM_VALUES structure, the server MUST write the number of PRINTER_ENUM_VALUES structures that were written to BUFFER into the variable pointed to by the **pnEnumValues** parameter.
- The server MUST return 0 for success or a nonzero error code.

With the exception of diagnostic purposes, the server state as visible to the client through this or any other protocol MUST NOT change as a result of processing this call.

3.1.4.1.9 CONTAINER Parameters

This section specifies common [CONTAINER](#) parameters and related validation and processing requirements.

3.1.4.1.9.1 DEVMODE_CONTAINER Parameters

pDevMode: This parameter MUST be a non-null pointer to a [DEVMODE_CONTAINER \(section 2.2.1.2.1\)](#) structure.

pDevModeContainer: This parameter is synonymous with **pDevMode**.

The individual method sections include the following parameter validation steps by reference:

- Verify that **pDevModeContainer** is not null.
- Verify that the **pDevMode** member of the **DEVMODE_CONTAINER** that is pointed to by **pDevModeContainer** is null or that the [DEVMODE](#) that is pointed to by the **pDevMode** member is valid, which means that the total size that is specified in DEVMODE MUST be less than or equal to the number of bytes that are specified by the value of the **cbBuf** member of the **DEVMODE_CONTAINER**.
- Verify that all members of the DEVMODE structure comply with the constraints defined in [2.2.2.1](#).
- Unless noted otherwise, if any of the above validation steps fail, return [ERROR_INVALID_PARAMETER](#).

3.1.4.1.9.2 DOC_INFO_CONTAINER Parameters

pDocInfoContainer: This parameter MUST be a non-null pointer to a [DOC_INFO_CONTAINER \(section 2.2.1.2.2\)](#) structure that MUST specify the document to print.

The individual method sections include the following parameter validation steps by reference:

- Verify that **pDocInfoContainer** is not null.
- Verify that the value of the **Level** member in the **DOC_INFO_CONTAINER** is one.
- Verify that the **pDocInfo1** pointer in the **DOC_INFO_CONTAINER** is non-null.
- Verify that all members of the structure that is pointed to by the **pDocInfo1** pointer in the **DOC_INFO_CONTAINER** structure comply with the constraints defined in [2.2.1.2.2](#).

Unless noted otherwise, if any of the above validation steps fail, return [ERROR_INVALID_PARAMETER](#).

3.1.4.1.9.3 DRIVER_CONTAINER Parameters

pDriverContainer: This parameter MUST be a non-null pointer to a [DRIVER_CONTAINER](#) structure that MUST specify printer driver information. The value of the **Level** member of the **DRIVER_CONTAINER** (section 2.2.1.2.3) structure MUST be two, three, four, six, or eight.

The individual method sections include the following parameter validation steps by reference:

- Verify that **pDriverContainer** is not null.
- Verify that **pDriverContainer** points to a **DRIVER_CONTAINER** that specifies an appropriate level as defined in the referring method definition. If that verification fails, return [ERROR_INVALID_LEVEL](#).
- Verify that **pDriverContainer** contains a non-null pointer to a structure and that all members of that structure comply with the constraints that are defined in section [2.2.1.2.3](#).

- Unless noted otherwise, if any of the above validation steps fail, return [ERROR_INVALID_PARAMETER](#).

3.1.4.1.9.4 FORM_CONTAINER Parameters

pFormInfoContainer: This parameter MUST be a non-null pointer to a [FORM_CONTAINER \(section 2.2.1.2.4\)](#) structure that MUST specify form information.

The individual method sections include the following parameter validation steps by reference:

- Verify that **pFormInfoContainer** is not null.
- Verify **pFormInfoContainer** points to a **FORM_CONTAINER** that specifies an appropriate level as defined in the referring method definition. If that verification fails, return **ERROR_INVALID_LEVEL**.
- Verify that **pFormInfoContainer** contains a non-null pointer to a structure and that all members of that structure comply with the constraints that are defined in section [2.2.1.2.4](#).

Unless noted otherwise, if any of the above validation steps fail, return [ERROR_INVALID_PARAMETER](#).

3.1.4.1.9.5 PORT_CONTAINER Parameters

pPortContainer: This parameter MUST be a non-null pointer to [PORT_CONTAINER \(section 2.2.1.2.7\)](#) structure that MUST specify port information.

The individual method sections include the following parameter validation steps by reference:

- Verify that **pPortContainer** is not null.
- Verify that **pPortContainer** points to a **PORT_CONTAINER** that specifies an appropriate level as defined in the referring method definition, and if that verification fails, return **ERROR_INVALID_LEVEL**.
- Verify that **pPortContainer** contains a non-null pointer to a structure and that all members of that structure comply with the constraints defined in section [2.2.1.2.7](#).

Unless noted otherwise, if any of the above validation steps fail, return [ERROR_INVALID_PARAMETER](#).

3.1.4.1.9.6 PRINTER_CONTAINER Parameters

pPrinterContainer: This parameter MUST be a non-null pointer to a [PRINTER_CONTAINER \(section 2.2.1.2.9\)](#) structure that MUST specify printer information. The **Level** member of the **PRINTER_CONTAINER** MUST be between zero and nine, inclusive. When the **Level** member is two, the **Status**, **cJobs**, and **AveragePPM** members of the [PRINTER_INFO 2](#) structure MUST be set to zero by the caller and MUST be ignored on receipt. For more information about [PRINTER_INFO](#) structures, see section [2.2.1.10](#).

The individual method sections include the following parameter validation steps by reference:

- Verify that **pPrinterContainer** is not null.
- Verify **pPrinterContainer** points to a **PRINTER_CONTAINER** specifying an appropriate level as defined in the referring method definition, and if that verification fails, return **ERROR_INVALID_LEVEL**.

- Verify that **pPrinterContainer** contains a non-null pointer to a structure and that all members of the structure pointed to comply with the constraints defined in [2.2.1.10](#), with the exception of [pServerName](#), which SHOULD be ignored.
- Unless noted otherwise, if any of the above validation steps fail, return [ERROR_INVALID_PARAMETER](#).

The individual method sections further include the following parameter processing steps by reference:

- If the value of the **Level** member specifies a [PRINTER_INFO](#) structure that contains a **pDevMode** member, replace the value of the **pDevMode** member with the **pDevMode** member of the [DEVMODE_CONTAINER](#) structure that is pointed to by the **pDevModeContainer** parameter of the method.
- If the value of the **Level** member specifies a [PRINTER_INFO](#) structure that contains a **pSecurity** member, replace the value of that **pSecurity** member with the **pSecurity** member of the [SECURITY_CONTAINER](#) structure that is pointed to by the **pSecurityContainer** parameter of the method.

3.1.4.1.9.7 SECURITY_CONTAINER Parameters

pSecurityContainer: This parameter MUST be a non-null pointer to a [SECURITY_CONTAINER](#) (section [2.2.1.2.13](#)) structure that MUST specify security information and components. The created printer MUST allow security access based on this information. [<78>](#)

The individual method sections include the following parameter validation steps by reference:

- Verify that **pSecurityContainer** is not null.
- Verify that the **pSecurity** member of the **SECURITY_CONTAINER** structure is null or that it points to a well-formed [SECURITY_DESCRIPTOR](#) in self-relative form (as specified in [\[MS-DTYP\]](#)).
- Unless noted otherwise, if any of the above validation steps fail, return [ERROR_INVALID_PARAMETER](#).

3.1.4.1.9.8 SPLCLIENT_CONTAINER Parameters

pClientInfo: This parameter MUST be a non-null pointer to an [SPLCLIENT_CONTAINER](#) (section [2.2.1.2.14](#)) structure that MUST specify client information. The level value of the **CONTAINER** MUST be one. [<79>](#)

The individual method sections include the following parameter validation steps by reference:

- Verify that **pClientInfo** is null or it points to an **SPLCLIENT_CONTAINER** that contains a non-null pointer to a structure and that all members of that structure comply with the constraints defined in [2.2.1.2.14](#).
- Unless otherwise noted, if the above validation steps fail, return [ERROR_INVALID_PARAMETER](#).

3.1.4.1.10 PRINTER_HANDLE Parameters

hPrinter: This parameter MUST specify an RPC context handle to an object managed by the server. RPC context handle are specified in [\[C706\]](#) section [2.3](#). The individual method sections describe

which methods **MUST** be used to obtain the handle and which of the object kinds (printer object, server object, port object, or job object) are acceptable.

The individual method sections include the following parameter validation steps by reference:

- Verify that **hPrinter** is an RPC context handle to an object managed by the server.
- Verify that **hPrinter** is an RPC context handle to an appropriate object as defined in the referring method definition.
- Verify that **hPrinter** has been opened with access rights deemed appropriate for the operation by server policy.
- Unless noted otherwise, if the above validation steps fail, return **ERROR_INVALID_HANDLE** as specified in [\[MS-ERREF\]](#).

3.1.4.1.11 Standard Parameter Validation

The implementation **MUST** apply the following validation rules to all parameters unless more specific statements appear in the individual method sections.

Term used to describe parameter	Required validation
X MUST be a non-null pointer to a string.	Verify that X is not null. If that verification fails, return ERROR_INVALID_PARAMETER .
X MUST be A.	Verify that X is A. If that verification fails, return ERROR_INVALID_PARAMETER .
X MUST be a value from A through B, inclusive.	Verify that X is a value that is greater than or equal to A and less than or equal to B. If that verification fails, return ERROR_INVALID_PARAMETER .
X MUST NOT be A.	Verify that X is a value that is not A. If that verification fails, return ERROR_INVALID_PARAMETER .
X MUST be one of <list>.	Verify that X is a value that is a member of <list>. If that verification fails, return ERROR_INVALID_PARAMETER .
X MUST be the result of a bitwise OR of zero or more of the flags in <list>.	If <list> contains the statement "All other bits MUST be zero", verify that the only bits that are set are those that are specified in <list>. If that verification fails, return ERROR_INVALID_PARAMETER .
X MUST be the result of a bitwise OR of one or more of the flags in <list>.	Verify that at least one of the bit flags from <list> is set and if that verification fails, return ERROR_INVALID_PARAMETER . If <list> contains the statement "All other bits MUST be zero", verify that the only bits that are set are the bits that are specified in <list>. If that verification fails, return ERROR_INVALID_PARAMETER .

3.1.4.2 Printer Management and Discovery Methods

This section specifies methods for discovering and obtaining access to supported printers.

Method	Description
RpcEnumPrinters	RpcEnumPrinters enumerates available printers, print servers,

Method	Description
	domains, or print providers. Opnum 0
RpcOpenPrinter	RpcOpenPrinter retrieves a printer handle, port handle, job handle, or print server handle. Opnum 1
RpcAddPrinter	RpcAddPrinter adds a printer to the list of supported printers for a specified server. Opnum 5
RpcDeletePrinter	RpcDeletePrinter deletes the specified printer object. Opnum 6
RpcSetPrinter	RpcSetPrinter sets the data for a specified printer or sets the state of the specified printer by pausing or resuming printing or clearing all print jobs. Opnum 7
RpcGetPrinter	RpcGetPrinter retrieves information about a specified printer. Opnum 8
RpcGetPrinterData	RpcGetPrinterData retrieves printer configuration data for a printer or print server. Opnum 26
RpcSetPrinterData	RpcSetPrinterData sets the configuration data for a printer or print server. Opnum 27
RpcClosePrinter	RpcClosePrinter closes a handle to a printer object, server object, job object, or port object. Opnum 29
RpcCreatePrinterIC	RpcCreatePrinterIC called by the graphics device interface (GDI) to create an information context for a printer. Opnum 40
RpcPlayGdiScriptOnPrinterIC	RpcPlayGdiScriptOnPrinterIC returns identifying information for fonts available for printing to a printer object. Opnum 41
RpcDeletePrinterIC	RpcDeletePrinterIC deletes a printer information context (IC). Opnum 42
RpcResetPrinter	RpcResetPrinter resets the data type and device mode values to use for printing documents submitted by the RpcStartDocPrinter method (see section 3.1.4.9.1). Opnum 52
RpcOpenPrinterEx	RpcOpenPrinterEx retrieves a printer handle, port handle, job handle, or print server handle. Opnum 69

Method	Description
RpcAddPrinterEx	RpcAddPrinterEx installs a printer on the print server. Opnum 70
RpcEnumPrinterData	RpcEnumPrinterData enumerates configuration data for a specified printer. Opnum 72
RpcDeletePrinterData	RpcDeletePrinterData deletes specified configuration data for a printer. Opnum 73
RpcSetPrinterDataEx	RpcSetPrinterDataEx sets the configuration data for a printer or print server. This method extends RpcSetPrinterData (see section 3.1.4.2.8), but by additionally allowing the caller to specify the registry key under which to store the data. Opnum 77
RpcGetPrinterDataEx	RpcGetPrinterDataEx retrieves configuration data for the specified printer or print server. This method extends RpcGetPrinterData (see section 3.1.4.2.7) and can retrieve values sorted under a specified key by RpcSetPrinterDataEx (see section 3.1.4.2.18). Opnum 78
RpcEnumPrinterDataEx	RpcEnumPrinterDataEx enumerates all value names and data for a specified printer and key. This method extends RpcEnumPrinterData (see section 3.1.4.2.16) by retrieving several values in a single call. Opnum 79
RpcEnumPrinterKey	RpcEnumPrinterKey enumerates the subkeys of a specified key for a specified printer. Printer data is stored in the registry. Opnum 80
RpcDeletePrinterDataEx	RpcDeletePrinterDataEx deletes a specified value from a printer's configuration data, which consists of a set of named and typed values stored in a hierarchy of registry keys. Opnum 81
RpcDeletePrinterKey	RpcDeletePrinterKey deletes a specified key and all of its subkeys for a specified printer. Opnum 82
RpcAddPerMachineConnection	RpcAddPerMachineConnection stores the print server name and the name of the binary executable used as a provider for a particular connection. Opnum 85
RpcDeletePerMachineConnection	RpcDeletePerMachineConnection deletes information about a server and connection provider. Opnum 86
RpcEnumPerMachineConnections	RpcEnumPerMachineConnections enumerates each connection and copies PRINTER_INFO_4 structures (see section 2.2.1.10.5)

Method	Description
	for all the per-machine connections into the buffer pPrinterEnum. Opnum 87
RpcSendRecvBidiData	RpcSendRecvBidiData sends and receives bidirectional data. This method is used to communicate with port monitors that support such data. Opnum 97

3.1.4.2.1 RpcEnumPrinters (Opnum 0)

RpcEnumPrinters enumerates available printers, print servers, domains, or print providers.

```

DWORD RpcEnumPrinters(
    [in] DWORD Flags,
    [in, string, unique] STRING_HANDLE Name,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check]
        BYTE* pPrinterEnum,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded,
    [out] DWORD* pcReturned
);

```

Flags: The value of this parameter specifies the types of print objects that the method **MUST** enumerate. The value of this parameter **MUST** be the result of a bitwise OR of one or more of the following values:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	G	F	E	D	C	B	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Where the bits are defined as:

Value	Description
A PRINTER_ENUM_DEFAULT	Default printer object enumeration. <80>
B PRINTER_ENUM_LOCAL	Enumerate local printer objects.
C PRINTER_ENUM_CONNECTIONS	Enumerates the printer connection previously added through RpcAddPerMachineConnection .
D PRINTER_ENUM_NAME	Enumerate printers on the server, domain, or print provider specified by Name . If Name is NULL or points to an empty string, and Level is one, enumerate available print providers.
E	Enumerate network printers and other print servers that are in the same domain as the print server. Level MUST be set to one

Value	Description
PRINTER_ENUM_REMOTE	if this flag is used.
F PRINTER_ENUM_SHARED	Only enumerate printers with the shared attribute set. This flag MUST be combined with one of the other flags.
G PRINTER_ENUM_NETWORK	Enumerate network printers that are in the same domain as the print server. Level MUST be set to one if this flag is used.

Name: This parameter MUST adhere to the parameter specification in [Print Server Name Parameters](#), section [3.1.4.1.2](#).

Level: This value refers to the level of printer information structure, as follows:

Value	Meaning
0x00000000	Corresponds to _PRINTER_INFO_STRESS, specified in section 2.2.2.10.1 .
0x00000001	Corresponds to _PRINTER_INFO_1, specified in section 2.2.2.10.2 .
0x00000002	Corresponds to _PRINTER_INFO_2, specified in section 2.2.2.10.3 .
0x00000004	Corresponds to _PRINTER_INFO_4, specified in section 2.2.2.10.5 .
0x00000005	Corresponds to _PRINTER_INFO_5, specified in section 2.2.2.10.6 .

pPrinterEnum: This parameter MAY be null if cbBuf equals zero; otherwise, it MUST be a non-null pointer to [BUFFER](#), as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

BUFFER TYPE: _PRINTER_INFO.

cbBuf: This parameter MUST adhere to the parameter specifications in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specifications in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcReturned: This parameter MUST adhere to the parameter specifications in INFO Structures Query Parameters, section [3.1.4.1.5](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST [<81>](#) validate parameters as follows:

- Perform validation steps as specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform validation steps as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- If the **PRINTER_ENUM_NETWORK** or **PRINTER_ENUM_REMOTE** flags are set, verify that the value of the **Level** parameter is one and if that verification fails, return **ERROR_INVALID_LEVEL**, as specified in [\[MS-ERREF\]](#).

If parameter validation fails, the server MUST fail the operation immediately and return a non-zero error response to the client. [<82>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- If the value of the **Level** parameter is one and the **PRINTER_ENUM_NETWORK** bit is set in the **Flags** parameter:
 - Enumerate all printers [<83>](#) in the "List of Known Printers" as specified in section [3.1.1](#).
If the server does not maintain a list of known printers, or if the list of known printers has not had at least one entry for an implementation-specific period of time, return **ERROR_CAN_NOT_COMPLETE**, as specified in [MS-ERREF].
- Otherwise:
 - Enumerate all printers [<84>](#) on the server or print provider that comply with the value of the **Flags** parameter.
- Using the enumerated objects, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Return the status of the operation. [<85>](#)

3.1.4.2.2 RpcOpenPrinter (Opnum 1)

RpcOpenPrinter retrieves a printer handle, port handle, job handle, or print server handle.

```
DWORD RpcOpenPrinter(  
    [in, string, unique] STRING_HANDLE pPrinterName,  
    [out] PRINTER_HANDLE* pHandle,  
    [in, string, unique] wchar_t* pDatatype,  
    [in] DEVMODE_CONTAINER* pDevModeContainer,  
    [in] DWORD AccessRequired  
);
```

pPrinterName: This parameter MUST adhere to the parameter specification in [Printer Name Parameters](#), section [3.1.4.1.1](#). This parameter MUST specify a printer connection, printer object, server object, job object, or port object.

pHandle: This parameter MUST be a non-null pointer to a variable that MUST receive the printer Remote Procedure Call (RPC) context handle to the object specified with **pPrinterName**. RPC context handles are specified in [\[C706\]](#) section [2.3](#).

pDatatype: This parameter MUST adhere to the parameter specification in [Datatype Name Parameters](#), section [3.1.4.1.3](#).

pDevModeContainer: This parameter MUST adhere to the parameter specification in [DEVMODE CONTAINER Parameters](#), section [3.1.4.1.9.1](#).

AccessRequired: The value of this parameter MUST specify the access level that the user requires to interact with the object for which the handle is being opened. If no specific access level is requested, the server MUST assume generic read access level. For details on access values, see section [2.2.3.4](#). For rules governing access values, see section [2.2.4.3](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Printer Name Parameters, section [3.1.4.1.1](#).
- Perform the validation steps that are specified in Datatype Name Parameters, section [3.1.4.1.3](#).
- Perform the validation steps that are specified in DEVMODE_CONTAINER Parameters, section [3.1.4.1.9.1](#).
- Verify that the client issuing the call has authorization equivalent to the value of the *AccessRequired* parameter.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<86>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Locate the printer, server, job, or port that corresponds to the request.
- Create an implementation-specific representation of the printer, server, job, or port ("the object") that MUST include:
 - An RPC handle, which is a snapshot of the printer, server, job, or port data that is specific to this instance of the invocation.
 - The data from **pDataType** and **pDevMode**, if they were not null.
 - All other relevant, implementation-specific data required to process all other protocol methods passing in a [PRINTER_HANDLE](#).
- Store the RPC handle for the object in the variable pointed to by **pHandle**.
- Increment the reference count of the object to prevent deletion.
- Return the status of the operation. [<87>](#)

3.1.4.2.3 RpcAddPrinter (Opnum 5)

RpcAddPrinter adds a printer to the list of supported printers for a specified server.

```

DWORD RpcAddPrinter(
    [in, string, unique] STRING_HANDLE pName,
    [in] PRINTER_CONTAINER* pPrinterContainer,
    [in] DEVMODE_CONTAINER* pDevModeContainer,
    [in] SECURITY_CONTAINER* pSecurityContainer,
    [out] PRINTER_HANDLE* pHandle
);

```

pName: This parameter MUST adhere to the parameter specification in [Print Server Name Parameters](#), section [3.1.4.1.2](#).

pPrinterContainer: This parameter MUST adhere to the parameter specification in [PRINTER_CONTAINER Parameters](#), section [3.1.4.1.9.6](#). The **Level** member of the **PRINTER_CONTAINER** MUST be one or two.

pDevModeContainer: This parameter MUST adhere to the parameter specification in [DEVMODE_CONTAINER Parameters](#), section [3.1.4.1.9.1](#).

pSecurityContainer: This parameter MUST adhere to the parameter specification in [SECURITY_CONTAINER Parameters](#), section [3.1.4.1.9.7](#).

pHandle: This parameter MUST be a non-null pointer to a variable that MUST receive the printer RPC context handle to the printer object added. RPC context handles are specified in [\[C706\]](#) section [2.3](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a non-zero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform validation steps as specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform validation steps as specified in PRINTER_CONTAINER Parameters, section [3.1.4.1.9.6](#).
- Perform validation steps as specified in DEVMODE_CONTAINER Parameters, section [3.1.4.1.9.1](#).
- Perform validation steps as specified in SECURITY_CONTAINER Parameters, section [3.1.4.1.9.7](#).
- If the value of the **Level** member of the **PRINTER_CONTAINER** is one:
 - If the server does not maintain a "List of Known Printers", return **ERROR_PRINTER_ALREADY_EXISTS**, as specified in [\[MS-ERREF\]](#).
- If the **Level** member of the **PRINTER_CONTAINER** is two:
 - Verify that the printer driver specified in the [PRINTER_INFO](#) that is pointed to by the **pointer** member of the **PRINTER_CONTAINER** pointed to by **pPrinterContainer** already exists in the system, and if that verification fails, return **ERROR_UNKNOWN_PRINTER_DRIVER**, as specified in [\[MS-ERREF\]](#).
 - Verify that the port specified in the PRINTER_INFO that is pointed to by the **pointer** member of the **PRINTER_CONTAINER** pointed to by **pPrinterContainer** already exists in the system, and if that verification fails, return **ERROR_UNKNOWN_PORT**, as specified in [\[MS-ERREF\]](#).
 - Verify that the print processor specified in the PRINTER_INFO that is pointed to by the **pointer** member of the **PRINTER_CONTAINER** pointed to by **pPrinterContainer** already exists in the system, and if that verification fails, return **ERROR_UNKNOWN_PRINTPROCESSOR**, as specified in [\[MS-ERREF\]](#).
 - Verify [<88>](#) that the printer with the name specified in the PRINTER_INFO that is pointed to by the **pointer** member of the **PRINTER_CONTAINER** pointed to by **pPrinterContainer** does not already exist in the system, and if that verification fails, return **ERROR_PRINTER_ALREADY_EXISTS**, as specified in [\[MS-ERREF\]](#).

If parameter validation fails, the server MUST [fail the operation immediately and return a nonzero error response to the client](#).

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Perform **PRINTER_CONTAINER** parameter processing steps as specified in PRINTER_CONTAINER Parameters, section [3.1.4.1.9.6](#).
- If the value of the **Level** member of the **PRINTER_CONTAINER** is one:
 - If the [PRINTER_ATTRIBUTE_SHARED](#) bit is set in the **Flags** member of the [PRINTER_INFO_1](#) structure, add the printer [to the "List of Known Printers"](#) as specified in section [3.1.1](#).
 - If PRINTER_ATTRIBUTE_SHARED bit is not set in the **Flags** member of the **PRINTER_INFO_1** structure, remove the printer from the network browse list.
 - Store null in the output parameter pointed to by **pHandle**.
 - Return **ERROR_PRINTER_ALREADY_EXISTS**, as specified in [MS-ERREF].

Note: An error return code is required by **RPC** because null was stored to the output parameter pointed to by **pHandle**.

- If the **Level** member of the **PRINTER_CONTAINER** is two:
 - Instead of failing the validation steps for missing printer driver, port, and print processor, the server MAY create the required printer driver, port, and print processor if they do not exist in the system. [91](#)
 - Create the printer object and assign to it the security descriptor from the [SECURITY_CONTAINER](#) that is pointed to by **pSecurityContainer** parameter.
 - Add the printer to the list of printers.
 - Create a session that includes:
 - An **RPC handle**
 - A snapshot of the printer data specific to this instance of the printer invocation
 - The data from the [DEVMODE](#) that is contained in the [DEVMODE_CONTAINER](#) pointed to by the **pDevModeContainer** parameter if it is not null
 - Store the **RPC handle** for the session in the output parameter pointed to by **pHandle**.
 - Increment the printer's reference count to prevent deletion.
 - If there are any clients that are registered for notifications on the server object change, a notification MUST be sent to those clients.
- Return the status of the operation. [92](#)

3.1.4.2.4 RpcDeletePrinter (Opnum 6)

RpcDeletePrinter is a method that deletes the specified printer object.

```
DWORD RpcDeletePrinter(
```

```
[in] PRINTER_HANDLE hPrinter
);
```

hPrinter: This parameter MUST specify a handle to a printer object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps as specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- If any jobs are pending on the printer, use the implementation-specific policy [<93>](#) to determine if a delete operation can be made pending or if an error should be returned.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<94>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Mark the printer object as "Delete Pending".
- Modify the list of printers in the system to exclude the deleted printer for any subsequent calls to [RpcEnumPrinters](#), [RpcOpenPrinter](#), [RpcOpenPrinterEx](#), [RpcAddJob](#), and [RpcStartDocPrinter](#). Clients that already have a valid handle to the same printer object MAY continue using the printer object for all operations except [RpcAddJob](#) and [RpcStartDocPrinter](#).
- If any clients have registered for notifications of the server object change, a notification MUST be broadcast to them.
- Return the status of the operation. [<95>](#)

3.1.4.2.5 RpcSetPrinter (Opnum 7)

RpcSetPrinter sets the data for a specified printer or sets the state of the specified printer by pausing or resuming printing or clearing all print jobs.

```
DWORD RpcSetPrinter(
    [in] PRINTER_HANDLE hPrinter,
    [in] PRINTER_CONTAINER* pPrinterContainer,
    [in] DEVMODE_CONTAINER* pDevModeContainer,
    [in] SECURITY_CONTAINER* pSecurityContainer,
    [in] DWORD Command
);
```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pPrinterContainer: This parameter MUST adhere to the parameter specification in [PRINTER_CONTAINER Parameters](#), section [3.1.4.1.9.6](#).

pDevModeContainer: This parameter MUST adhere to the parameter specification in [DEVMODE_CONTAINER Parameters](#), section [3.1.4.1.9.1](#).

pSecurityContainer: This parameter MUST adhere to the parameter specification in [SECURITY_CONTAINER Parameters](#), section [3.1.4.1.9.7](#).

Command: If the value of this parameter is zero, the **PrinterInfo** member of the [PRINTER_CONTAINER](#) that is pointed to by the **pPrinterContainer** parameter MUST contain a non-null pointer to a [PRINTER_INFO](#) structure that **RpcSetPrinter** can use. See section [2.2.1.10.1](#) for details. If the value of this parameter is not zero, the value of the **Level** member of the **PRINTER_CONTAINER** that is pointed to by **pPrinterContainer** MUST be zero and the value MUST be a constant value from the following table that specifies the action to perform.

Printer Control Name/Value	Meaning
PRINTER_CONTROL_PAUSE 0x00000001	Pauses the printer object.
PRINTER_CONTROL_RESUME 0x00000002	Resumes a paused printer object.
PRINTER_CONTROL_PURGE 0x00000003	Deletes all print jobs queued for the printer object.

Return Values:

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in [PRINTER_HANDLE Parameters](#), section [3.1.4.1.10](#).
- Perform the validation steps that are specified in [PRINTER_CONTAINER Parameters](#), section [3.1.4.1.9.6](#).
- Verify that the information that is provided in the **PRINTER_CONTAINER** that is pointed to by the **pPrinterContainer** parameter is consistent with the value in **Command**, according to the following table:

Command	Level in PRINTER_CONTAINER
0x00000000	MUST be a decimal number from zero to nine, inclusive.
0x00000001	MUST be zero.

Command	Level in PRINTER_CONTAINER
0x00000002	MUST be zero.
0x00000003	MUST be zero.

- Perform validation steps as specified in **DEVMODE_CONTAINER** Parameters, section [3.1.4.1.9.1](#).
- Perform validation steps as specified in **SECURITY_CONTAINER** Parameters, section [3.1.4.1.9.7](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<96>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- If **hPrinter** specifies a server object, the server MUST only apply the **SECURITY_CONTAINER** parameter to set the print server's security descriptor, and MUST not perform the remaining processing steps that follow.
- Perform **PRINTER_CONTAINER** parameter processing steps that are specified in **PRINTER_CONTAINER** Parameters, section [3.1.4.1.9.6](#).
- Perform the operation from the following table that corresponds to the value of the **Command** parameter.

Command parameter values	Operation that MUST be performed
0x00000000 No command	Update the printer configuration using the settings in pPrinterContainer .
0x00000001 Pause the printer	Temporarily suspend sending data to the printer without changing the state of any jobs associated with the printer. Clients MAY continue adding data to the job.
0x00000002 Resume the printer	Resume sending data to the printer without changing the state of any jobs associated with the printer.
0x00000003 Purge the printer	Remove all jobs that are currently associated with the printer and mark them as having failed to print.

- If any clients registered for notifications of the printer object change, a notification MUST be broadcast to them.
- Return the status of the operation. [<97>](#)

3.1.4.2.6 **RpcGetPrinter (Opnum 8)**

RpcGetPrinter retrieves information about a specified printer.

```
DWORD RpcGetPrinter(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check]
```

```

    BYTE* pPrinter,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded
);

```

hPrinter: This parameter MUST specify a handle to a printer object<98> that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

Level: This value refers to the level of printer information structure, as follows:

Value	Meaning
0x00000000	Corresponds to _PRINTER_INFO_STRESS, specified in section 2.2.1.10.1 .
0x00000001	Corresponds to _PRINTER_INFO_1, specified in section 2.2.2.10.2 .
0x00000002	Corresponds to _PRINTER_INFO_2, specified in section 2.2.2.10.3 .
0x00000003	Corresponds to _PRINTER_INFO_3, specified in section 2.2.2.10.4 .
0x00000004	Corresponds to _PRINTER_INFO_4, specified in section 2.2.2.10.5 .
0x00000005	Corresponds to _PRINTER_INFO_5, specified in section 2.2.2.10.6 .
0x00000006	Corresponds to _PRINTER_INFO_6, specified in section 2.2.2.10.7 .
0x00000007	Corresponds to _PRINTER_INFO_7, specified in section 2.2.2.10.8 .
0x00000008	Corresponds to _PRINTER_INFO_8, specified in section 2.2.2.10.9 .
0x00000009	Corresponds to _PRINTER_INFO_9, specified in section 2.2.2.10.10 .

pPrinter: This parameter MUST be a non-null pointer to [BUFFER](#) as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

BUFFER TYPE: _PRINTER_INFO.

cbBuf: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in [PRINTER_HANDLE Parameters](#), section [3.1.4.1.10](#).

- Perform the validation steps that are specified in INFO Structures Query Parameters, section [3.1.4.1.5.<99>](#)

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client.[<100>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Using information about the printer, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Return the status of the operation.[<101>](#)

3.1.4.2.7 RpcGetPrinterData (Opnum 26)

RpcGetPrinterData retrieves printer configuration data for a printer or print server.

```
DWORD RpcGetPrinterData(
    [in] PRINTER_HANDLE hPrinter,
    [in, string] wchar_t* pValueName,
    [out] DWORD* pType,
    [out, size_is(nSize)] BYTE* pData,
    [in] DWORD nSize,
    [out] DWORD* pcbNeeded
);
```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pValueName: This parameter MUST be a non-null pointer to a string that MUST identify the data to get. For rules governing value names, see section [2.2.4.16](#).

For print servers, this string is one of the predefined strings listed in [Server Handle Key Values](#), section [2.2.3.8](#). For printer objects, the following value name has a special meaning:

Value	Meaning
"ChangeID"	<p>In the buffer pointed to by pData, return a DWORD value representing a change identifier. This identifier MUST be set by the print server to a new unique value each time printer information changes. The client SHOULD use the identifier to decide if it has stale information about a printer object, in which case the client SHOULD call RpcGetPrinter or RpcGetPrinterData to update its view of the printer object.</p> <p>Note Only the fact that the value changes is significant; the value itself is arbitrary.</p>

pType: This parameter MUST adhere to the parameter specification in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

pData: This parameter MAY be null if **nSize** equals zero; otherwise, it MUST be a non-null pointer to BUFFER as specified in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

nSize: This parameter MUST adhere to the parameter specification in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Perform the validation steps that are specified in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).
- Additional validation MAY [<102>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a non-zero error response to the client. [<103>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- With the data identified by **pValueName**, perform the processing and response steps [<104>](#) specified in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).
- Return the status of the operation. [<105>](#)

3.1.4.2.8 RpcSetPrinterData (Opnum 27)

RpcSetPrinterData sets the configuration data for a printer or print server.

```
DWORD RpcSetPrinterData(  
    [in] PRINTER_HANDLE hPrinter,  
    [in, string] wchar_t* pValueName,  
    [in] DWORD Type,  
    [in, size_is (cbData)] BYTE* pData,  
    [in] DWORD cbData  
);
```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pValueName: This parameter MUST be a non-null pointer to a string that MUST identify the data to set. For rules governing value names, see section [2.2.4.16.<106>](#)

Type: The value of this parameter MUST specify the type value for data pointed to by the **pData** parameter. The value SHOULD be one of the possible type codes defined by type values in section [2.2.3.9](#). For rules governing registry type values, see section [2.2.4.9](#).

pData: This parameter SHOULD be a non-null pointer to an array of bytes that MUST contain the printer configuration data. The type of the data in the buffer is specified by the **Type** parameter.

cbData: The value of this parameter MUST be the size, in bytes, of the **pData** array. This value SHOULD NOT be zero.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Additional validation MAY [<107>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<108>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Set the printer data [<109>](#) associated with **pValueName** to the data pointed to by **pData**.
- Return the status of the operation. [<110>](#)

3.1.4.2.9 RpcClosePrinter (Opnum 29)

RpcClosePrinter closes a handle to a printer object, server object, job object, or port object.

```
DWORD RpcClosePrinter(  
    [in, out] PRINTER_HANDLE* phPrinter  
);
```

phPrinter: This parameter MUST be a non-null pointer to a handle to a printer object, server object, job object, or port object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<111>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- If the object is a printer object, and [RpcStartDocPrinter](#) has been called without a matching [RpcEndDocPrinter](#), the same processing as for **RpcEndDocPrinter** MUST occur.

- If the object is a printer object and jobs have been added using [RpcAddJob](#) but have not been scheduled using [RpcScheduleJob](#), they MUST be scheduled now.
- If there is an active notification context associated with the object, as a result of the client not calling [RpcFindClosePrinterChangeNotification](#), the server MUST close the notification context now by calling the client's [RpcReplyClosePrinter](#) method.
- Free any internal state that is associated with the handle that is pointed to by **phPrinter**.
- Store null in the variable that is pointed to by **phPrinter**.
- Decrement the reference count on that object.
- If the object is a printer object marked as "Delete Pending", and the usage count is zero, the following steps MUST be performed:
 - Handle any pending jobs in an implementation-specific manner.
 - Clear references to this printer from any other data structures.
 - Delete the printer object.
- Return the status of the operation. [<112>](#)

3.1.4.2.10 RpcCreatePrinterIC (Opnum 40)

RpcCreatePrinterIC is called by the graphics device interface (GDI) to create an information context for a specified printer.

```
DWORD RpcCreatePrinterIC(
    [in] PRINTER_HANDLE hPrinter,
    [out] GDI_HANDLE* pHandle,
    [in] DEVMODE_CONTAINER* pDevModeContainer
);
```

hPrinter: This parameter MUST be a handle to a printer object (section [2.2.1.1.3](#)), which MUST have been opened using [RpcAddPrinter](#) (section [3.1.4.2.3](#)), [RpcAddPrinterEx](#) (section [3.1.4.2.15](#)), [RpcOpenPrinter](#) (section [3.1.4.2.2](#)), or [RpcOpenPrinterEx](#) (section [3.1.4.2.14](#)).

pHandle: This parameter MUST be a non-null pointer to a printer information context handle (section [2.2.1.1.1](#)).

pDevModeContainer: This parameter MUST adhere to the [DEVMODE_CONTAINER Parameters](#) (section [3.1.4.1.9.1](#)) specification.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in [PRINTER_HANDLE Parameters](#) (section [3.1.4.1.10](#)).

- Perform the validation steps that are specified in `DEVMODE_CONTAINER` Parameters.

If parameter validation fails, the server **MUST** fail the operation immediately and return a nonzero error response to the client. [<113>](#)

Otherwise, the server **MUST** process the message and compose a response to the client as follows:

- Perform implementation-specific steps to create the appropriate printer information context.
- Store an **RPC handle** associated with the information context in `pHandle`.
- Return the status of the operation. [<114>](#)

Except for diagnostic purposes, the server state, as visible to the client through this or any other protocol, **MUST NOT** change as a result of processing this call.

3.1.4.2.11 `RpcPlayGdiScriptOnPrinterIC` (Opnum 41)

`RpcPlayGdiScriptOnPrinterIC` returns font information for a printer connection.

```
DWORD RpcPlayGdiScriptOnPrinterIC(
    [in] GDI_HANDLE hPrinterIC,
    [in, size_is(cIn)] BYTE* pIn,
    [in] DWORD cIn,
    [out, size_is(cOut)] BYTE* pOut,
    [in] DWORD cOut,
    [in] DWORD ul
);
```

hPrinterIC: This parameter **MUST** be a printer information context handle (section [2.2.1.1.1](#)), which **MUST** have been returned by [`RpcCreatePrinterIC` \(section 3.1.4.2.10\)](#).

pIn: The value of this parameter **SHOULD** be null.

cIn: The value of this parameter **SHOULD** be zero.

pOut: The value of this parameter **SHOULD** be interpreted as follows:

cOut	pOut
0x00000004	This parameter MUST be a non-null pointer to a DWORD that specifies the number of UNIVERSAL_FONT_ID structures (section 2.2.1.1.7) to be returned by the next call.
0x00000004 <	This parameter MUST be a non-null pointer to a buffer, which contains a DWORD that specifies the number of UNIVERSAL_FONT_ID structures that immediately follow.

cOut: The value of this parameter **MUST** be the size in bytes of the buffer pointed to by **pOut**.

ul: The value of this parameter **SHOULD** be zero.

Return value/code	Description
0x00000000	This method MUST return zero to indicate successful completion or a nonzero

Return value/code	Description
ERROR_SUCCESS	Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate the following:

- Verify that **hPrinterIC** is a valid printer information context handle.
- Verify that the value of the **pOut** parameter is not null.
- Verify the value of the **cOut** parameter according to the following steps:
 - If the value of the **cOut** parameter is less than 0x00000004, return **ERROR_NOT_ENOUGH_MEMORY**, as specified in [\[MS-ERREF\]](#).
 - If the value of the **cOut** parameter is 0x00000004, proceed.
 - If the value of the **cOut** parameter is greater than 0x00000004, it MUST be at least the result of multiplying the size of(**UNIVERSAL_FONT_ID**) by the number of fonts and then adding 0x00000004 to that product; otherwise, **ERROR_NOT_ENOUGH_MEMORY** SHOULD be returned, as specified in [\[MS-ERREF\]](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<115>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Query the fonts that are available on the server.
- Write the number of fonts to the DWORD variable that is pointed to by the **pOut** parameter.
- If **cOut** is greater than 0x00000004, copy the font information into the buffer location that is pointed to by the value of **pOut** plus 0x00000004.
- Return the status of the operation. [<116>](#)

Except for diagnostic purposes, the server state, as visible to the client through this or any other protocol, MUST NOT change as a result of processing this call.

3.1.4.2.12 RpcDeletePrinterIC (Opnum 42)

RpcDeletePrinterIC deletes a printer information context.

```
DWORD RpcDeletePrinterIC(
    [in, out] GDI_HANDLE* phPrinterIC
);
```

phPrinterIC: This parameter MUST be a non-null pointer to a printer information context handle (section [2.2.1.1.1](#)), which MUST have been returned by [RpcCreatePrinterIC \(section 3.1.4.2.10\)](#).

Return value/code	Description
0x00000000	This method MUST return zero to indicate successful completion or a nonzero

Return value/code	Description
ERROR_SUCCESS	Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST verify that the handle pointed to by the **phPrinterIC** parameter is associated with an information context.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<117>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Delete the printer information context.
- Store null in the variable pointed to by **phPrinterIC**.
- Return the status of the operation. [<118>](#)

Except for diagnostic purposes, the server state, as visible to the client through this or any other protocol, MUST NOT change as a result of processing this call.

3.1.4.2.13 RpcResetPrinter (Opnum 52)

RpcResetPrinter resets the data type and device mode values to use for printing documents submitted by the [RpcStartDocPrinter](#) method (see section [3.1.4.9.1](#)).

```
DWORD RpcResetPrinter(
    [in] PRINTER_HANDLE hPrinter,
    [in, string, unique] wchar_t* pDataatype,
    [in] DEVMODE_CONTAINER* pDevModeContainer
);
```

hPrinter: This parameter MUST specify a handle to a printer object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pDatatype: This parameter MUST adhere to the parameter specification in Datatype Name Parameters, section [3.1.4.1.3](#).

pDevModeContainer: This parameter MUST adhere to the parameter specification in DEVMODE_CONTAINER Parameters, section [3.1.4.1.9.1](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Perform the validation steps that are specified in Datatype Name Parameters, section [3.1.4.1.3](#).

- Perform the validation steps that are specified in DEVMODE_CONTAINER Parameters, section [3.1.4.1.9.1](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<119>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Update the default data type that is associated with the context for **hPrinter**.
- Update the default [DEVMODE](#) structure that is associated with the context for **hPrinter**.
- Return the status of the operation. [<120>](#)

3.1.4.2.14 RpcOpenPrinterEx (Opnum 69)

RpcOpenPrinterEx retrieves a printer handle, port handle, job handle, or print server handle.

```
DWORD RpcOpenPrinterEx(
    [in, string, unique] STRING_HANDLE pPrinterName,
    [out] PRINTER_HANDLE* pHandle,
    [in, string, unique] wchar_t* pDatatype,
    [in] DEVMODE_CONTAINER* pDevModeContainer,
    [in] DWORD AccessRequired,
    [in] SPLCLIENT_CONTAINER* pClientInfo
);
```

pPrinterName: This parameter MUST adhere to the parameter specification in Printer Name Parameters, section [3.1.4.1.1](#). It MUST specify a printer connection, printer object, server object, job object, or port object.

pHandle: This parameter MUST be a non-null pointer to a variable that MUST receive the printer RPC context handle to the object specified with **pPrinterName**. RPC context handles are specified in [\[C706\]](#) section [2.3](#).

pDatatype: This parameter MUST adhere to the parameter specification in Datatype Name Parameters, section [3.1.4.1.3](#).

pDevModeContainer: This parameter MUST adhere to the parameter specification in DEVMODE_CONTAINER Parameters, section [3.1.4.1.9.1](#).

AccessRequired: The value of this parameter MUST specify the access level that the client requires for interacting with the object to which the handle is being opened. If no specific access level is requested, the server assumes a generic read access level. The value MUST be one of the access values described in section [2.2.3.4](#). For rules governing access values, see section [2.2.4.3](#).

pClientInfo: This parameter MUST adhere to the parameter in SPLCLIENT_CONTAINER Parameters, section [3.1.4.1.9.8](#). The value of the **Level** member of the container that is pointed to by **pClientInfo** MUST be one.

Return value/code	Description
0x00000000	This method MUST return zero to indicate successful completion or a nonzero

Return value/code	Description
ERROR_SUCCESS	Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Printer Name Parameters, section [3.1.4.1.1](#).
- Perform the validation steps that are specified in Datatype Name Parameters, section [3.1.4.1.3](#).
- Perform the validation steps that are specified in DEVMODE_CONTAINER Parameters, section [3.1.4.1.9.1](#).
- Perform the validation steps that are specified in SPLCLIENT_CONTAINER Parameters, section [3.1.4.1.9.8](#).
- Verify that the client issuing the call has authorization equivalent to the value of the **AccessRequired** parameter.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<121>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Locate the printer, server, job, or port that corresponds to the request.
- Create an implementation-specific representation of the printer, server, job, or port ("the object") that MUST include:
 - An **RPC handle**, which is a snapshot of the printer, server, job, or port data that is specific to this instance of the invocation.
 - The data from **pDatatype** and **pDevModeContainer**, if they were not null.
 - All other relevant, implementation-specific data that is required to process all other protocol methods that pass in a [PRINTER_HANDLE](#).
 - The data from **pClientInfo**, if it is not null.
- Increment the object's reference count to prevent deletion.
- Store the **RPC handle** for the session in the output parameter **pHandle**.
- Return the status of the operation. [<122>](#)

3.1.4.2.15 RpcAddPrinterEx (Opnum 70)

RpcAddPrinterEx installs a printer on the print server.

```

DWORD RpcAddPrinterEx(
    [in, string, unique] STRING_HANDLE pName,
    [in] PRINTER_CONTAINER* pPrinterContainer,
    [in] DEVMODE_CONTAINER* pDevModeContainer,
    [in] SECURITY_CONTAINER* pSecurityContainer,
    [in] SPLCLIENT_CONTAINER* pClientInfo,
    [out] PRINTER_HANDLE* pHandle
  )

```

);

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pPrinterContainer: This parameter MUST adhere to the parameter specification in PRINTER_CONTAINER Parameters, section [3.1.4.1.9.6](#). The **Level** member of the **PRINTER_CONTAINER** MUST be one or two.

pDevModeContainer: This parameter MUST adhere to the parameter specification in DEVMODE_CONTAINER Parameters, section [3.1.4.1.9.1](#).

pSecurityContainer: This parameter MUST adhere to the parameter specification in SECURITY_CONTAINER Parameters, section [3.1.4.1.9.7](#).

pClientInfo: This parameter MUST adhere to the parameter specification in SPLCLIENT_CONTAINER Parameters, section [3.1.4.1.9.8](#).

pHandle: This parameter MUST be a non-null pointer to a variable that MUST receive the printer RPC context handle to the printer object added. RPC context handles are specified in [\[C706\]](#) section [2.3](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in PRINTER_CONTAINER Parameters, section [3.1.4.1.9.6](#).
- Perform the validation steps that are in DEVMODE_CONTAINER Parameters, section [3.1.4.1.9.1](#).
- Perform the validation steps that are specified in SECURITY_CONTAINER Parameters, section [3.1.4.1.9.7](#).
- Perform the validation steps that are specified in SPLCLIENT_CONTAINER Parameters, section [3.1.4.1.9.8](#).
- If the value of the **Level** member of the **PRINTER_CONTAINER** that is pointed to by the **pPrinterContainer** parameter is one, and if the server does not maintain a "List of Known Printers", return ERROR_PRINTER_ALREADY_EXISTS, as specified in [\[MS-ERREF\]](#).
- If the value of the **Level** member of the **PRINTER_CONTAINER** that is pointed to by the **pPrinterContainer** parameter is two:
 - Verify that the printer driver specified in the [PRINTER_INFO](#) that is pointed to by the **pointer** member of the **PRINTER_CONTAINER** pointed to by the **pPrinterContainer** parameter already exists in the system; if that verification fails, return ERROR_UNKNOWN_PRINTER_DRIVER, as specified in [\[MS-ERREF\]](#).

- Verify that the port specified in the **PRINTER_INFO** that is pointed to by the **pointer** member of the **PRINTER_CONTAINER** pointed to by the **pPrinterContainer** parameter already exists in the system; if that verification fails, return **ERROR_UNKNOWN_PORT**, as specified in [MS-ERREF].
- Verify that the print processor specified in the **PRINTER_INFO** that is pointed to by the **pointer** member of the **PRINTER_CONTAINER** pointed to by the **pPrinterContainer** parameter already exists in the system; if that verification fails, return **ERROR_UNKNOWN_PRINTPROCESSOR**, as specified in [MS-ERREF].
- Verify that the printer with the name specified in the **PRINTER_INFO** that is pointed to by the **pointer** member of the **PRINTER_CONTAINER** pointed to by the **pPrinterContainer** parameter does not already exist in the system; if that verification fails, return **ERROR_PRINTER_ALREADY_EXISTS**, as specified in [MS-ERREF].
- Additional validation MAY [<123>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<124>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Perform **PRINTER_CONTAINER** parameter processing steps as specified in **PRINTER_CONTAINER** Parameters, section [3.1.4.1.9.6](#).
- If the value of the **Level** member of the **PRINTER_CONTAINER** that is pointed to by the **pPrinterContainer** parameter is one:
 - If the [PRINTER_ATTRIBUTE_SHARED](#) bit is set in the **Flags** member of the **PRINTER_INFO** structure pointed to by the **pPrinterInfo1** member of the **PRINTER_CONTAINER** that is pointed to by the **pPrinterContainer** parameter, add the printer to the "List of Known Printers" as specified in Abstract Data Model, section [3.1.1.<125>](#)
 - If **PRINTER_ATTRIBUTE_SHARED** is not set in the **Flags** member of the **PRINTER_INFO** structure pointed to by the **pPrinterInfo1** member of the **PRINTER_CONTAINER** that is pointed to by the **pPrinterContainer** parameter, remove the printer from the network browse list.
 - Store null in the output parameter that is pointed to by **pHandle**.
 - Return **ERROR_PRINTER_ALREADY_EXISTS**, as specified in [MS-ERREF].

Note: An error return code is required by **RPC** because null was stored to the output parameter pointed to by **pHandle**.
- If the value of the **Level** member of the **PRINTER_CONTAINER** that is pointed to by the **pPrinterContainer** parameter is two:
 - Instead of failing the validation steps for missing printer driver, port, and print processor, the server MAY create the required printer driver, port, and print processor if they do not exist in the system. [<126>](#)
 - Create the printer object and assign to it the security descriptor from the [SECURITY_CONTAINER](#) that is pointed to by the **pSecurityContainer** parameter.
 - Add the printer to the list of printers.
 - Create a session that includes:

- An **RPC handle**.
- A snapshot of the printer data specific to this instance of the printer invocation.
- The data from [DEVMODE](#) that is contained in the [DEVMODE_CONTAINER](#) pointed to by the **pDevModeContainer** parameter if it is not null.
- The data from the [SPLCLIENT_CONTAINER](#) that is pointed to by the **pClientInfo** parameter if it is not null.
- Store the **RPC handle** for the session in the output parameter **pHandle**.
- Increment the printer's reference count to prevent deletion.
- If any clients are registered for notifications of the server object change, a notification **MUST** be broadcast to them.
- Return the status of the operation. [<127>](#)

3.1.4.2.16 RpcEnumPrinterData (Opnum 72)

RpcEnumPrinterData enumerates configuration data for a specified printer.

```

DWORD RpcEnumPrinterData(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD dwIndex,
    [out, size_is(cbValueName/sizeof(wchar_t))]
    wchar_t* pValueName,
    [in] DWORD cbValueName,
    [out] DWORD* pcbValueName,
    [out] DWORD* pType,
    [out, size_is(cbData)] BYTE* pData,
    [in] DWORD cbData,
    [out] DWORD* pcbData
);

```

hPrinter: This parameter **MUST** specify a handle to a printer object that **MUST** have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

dwIndex: The value of this parameter **MUST** specify the configuration data value to retrieve. The value **MUST** be greater than or equal to zero and less than the total number of values for the printer. The client **SHOULD** use **RpcEnumPrinterKeys** to determine the total number of values.

pValueName: This parameter **MAY** be null if **cbValueName** equals zero; otherwise, it **MUST** be a non-null pointer to a buffer that **MUST** receive a string specifying the name of the configuration data value. For rules governing value names, see section [2.2.4.16](#).

cbValueName: The value of this parameter **MUST** be the size, in bytes, of the buffer that is pointed to by the **pValueName** parameter.

pcbValueName: This parameter **MUST** be a non-null pointer to a variable that **MUST** receive the number of bytes stored in the buffer that is pointed to by the **pValueName** parameter.

pType: This parameter MUST adhere to the parameter specification in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

pData: This parameter MAY be null if **cbData** equals zero; otherwise, it MUST be a non-null pointer to BUFFER as specified in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

cbData: This parameter MUST adhere to the parameter specification in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

pcbData: This parameter MUST adhere to the parameter specification in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that the value of the **cbValueName** parameter is not smaller than the number of bytes required to hold the string that specifies the name of the value. If that verification fails, the server MUST update the variable that is pointed to by the **pcbValueName** parameter with the number of bytes required and return ERROR_MORE_DATA, as specified in [\[MS-ERREF\]](#).
- Verify that the value of the **dwIndex** parameter is greater than or equal to zero, and smaller than the total number of values for the printer.
- Perform the validation steps that are specified in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<128>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Store the name of the printer property in the string buffer that is pointed to by the **pValueName** parameter and store the length of the name stored in the variable that is pointed to by the **pcbValueName** parameter.
- Using the data identified by **pValueName**, [<129>](#) perform the processing and response steps specified in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).
- Return the status of the operation. [<130>](#)

3.1.4.2.17 RpcDeletePrinterData (Opnum 73)

RpcDeletePrinterData deletes specified configuration data for a printer.

```
DWORD RpcDeletePrinterData(  
    [in] PRINTER_HANDLE hPrinter,  
    [in, string] wchar_t* pValueName  
);
```

hPrinter: This parameter MUST specify a handle to a printer object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pValueName: This parameter MUST be a non-null pointer to the string that MUST specify the name of the configuration data value to delete. For rules governing value names, see section [2.2.4.16](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Additional validation MAY [<131>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<132>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Delete the printer data specified by **pValueName**. [<133>](#)
- Return the status of the operation. [<134>](#)

3.1.4.2.18 RpcSetPrinterDataEx (Opnum 77)

RpcSetPrinterDataEx sets the configuration data for a printer or print server. This method is similar to **RpcSetPrinterData** (see section [3.1.4.2.8](#)) but additionally allows the caller to specify the registry key under which to store the data.

```
DWORD RpcSetPrinterDataEx(  
    [in] PRINTER_HANDLE hPrinter,  
    [in, string] const wchar_t* pKeyName,  
    [in, string] const wchar_t* pValueName,  
    [in] DWORD Type,  
    [in, size_is(cbData)] BYTE* pData,  
    [in] DWORD cbData  
);
```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pKeyName: This parameter MAY be null if **hPrinter** is a handle to a server object; otherwise, it MUST be a non-null pointer to a string that MUST specify the key under which the value is to be set. For rules governing key names, see section [2.2.4.17](#).

pValueName: This parameter MUST be a non-null pointer to a string that MUST identify the data to set. For rules governing value names, see section [2.2.4.16](#).

Type: The value of this parameter MUST be a code that indicates the type of data that is pointed to by the **pData** parameter. The value SHOULD be one of the possible type codes defined by type values in section [2.2.3.9](#). For rules governing registry type values, see section [2.2.4.9](#).

pData: This parameter SHOULD be a non-null pointer to an array of bytes that MUST contain the printer configuration data. The type of the data in the buffer is specified by the **Type** parameter.

cbData: The value of this parameter MUST be the size, in bytes, of the array. This value SHOULD NOT be zero.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- If the **hPrinter** parameter is a handle to a printer object, verify that the **pKeyName** parameter points to a string.

Additional validation MAY [<135>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<136>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- If the **hPrinter** parameter is a handle to a printer object, store the data that is provided by **pData** with the type that is supplied by **Type**, in the printer data value that is identified by **pKeyName** and **pValueName**.
- If **hPrinter** is a handle to a server object, store the data that is provided by **pData** with the type that is supplied by **Type**, in the server data value that is identified by the **pValueName** parameter.
- Return the status of the operation. [<137>](#)

3.1.4.2.19 RpcGetPrinterDataEx (Opnum 78)

RpcGetPrinterDataEx retrieves configuration data for the specified printer or print server. This method is similar to [RpcGetPrinterData](#) (see section [3.1.4.2.7](#)) but also allows the caller to specify the registry key from which to retrieve the data.

```
DWORD RpcGetPrinterDataEx(  
    [in] PRINTER_HANDLE hPrinter,  
    [in, string] const wchar_t* pKeyName,  
    [in, string] const wchar_t* pValueName,  
    [out] DWORD* pType,  
    [out, size_is(nSize)] BYTE* pData,  
    [in] DWORD nSize,  
    [out] DWORD* pcbNeeded
```

);

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pKeyName: This parameter MAY be null if **hPrinter** is a handle to a server object; otherwise, it MUST be a non-null pointer to a string that MUST specify the key under which the value is to be queried. For rules governing key names, see section [2.2.4.17](#).

pValueName: This parameter MUST be a non-null pointer to a string that MUST identify the data to get. For rules governing value names, see section [2.2.4.16](#).

See **RpcGetPrinterData**, section [3.1.4.2.7](#), for further information on the interpretation of this value.

pType: This parameter MUST adhere to the parameter specification in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

pData: This parameter MAY be null if **nSize** equals zero; otherwise, it MUST be a non-null pointer to BUFFER, as specified in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

nSize: This parameter MUST adhere to the parameter specification in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- If **hPrinter** is a handle to a printer object, verify that **pKeyName** points to a string.
- Perform the validation steps that are specified in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).
- Additional validation MAY [<138>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<139>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- If the **hPrinter** parameter is a handle to a printer object, with the data identified by **pKeyName** and **pValueName**, perform the processing and response steps that are specified in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).

- If **hPrinter** is a handle to a server object, with the data that is identified by **pValueName**, perform the processing and response steps that are specified in Dynamically Typed Query Parameters, section [3.1.4.1.7](#).
- Return the status of the operation. [<140>](#)

3.1.4.2.20 RpcEnumPrinterDataEx (Opnum 79)

RpcEnumPrinterDataEx enumerates all value names and data for a specified printer and key. This method is similar to **RpcEnumPrinterData** (see section [3.1.4.2.16](#)) but also allows the caller to specify the registry key from which to enumerate the data, and allows retrieving several values in a single call.

```
DWORD RpcEnumPrinterDataEx(
    [in] PRINTER_HANDLE hPrinter,
    [in, string] const wchar_t* pKeyName,
    [out, size_is(cbEnumValues)] BYTE* pEnumValues,
    [in] DWORD cbEnumValues,
    [out] DWORD* pcbEnumValues,
    [out] DWORD* pnEnumValues
);
```

hPrinter: This parameter MUST specify a handle to a printer object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pKeyName: This parameter MUST be a non-null pointer to a string that MUST specify the key containing the values to enumerate. For rules governing key names, see section [2.2.4.17](#).

pEnumValues: This parameter MAY be null if **cbEnumValues** equals zero; otherwise, it MUST be a non-null pointer to [BUFFER](#) as specified in PRINTER_ENUM_VALUES Structures Query Parameters, section [3.1.4.1.8](#).

cbEnumValues: This parameter must adhere to the parameter specification in PRINTER_ENUM_VALUES Structures Query Parameters, section [3.1.4.1.8](#).

pcbEnumValues: This parameter must adhere to the parameter specification in PRINTER_ENUM_VALUES Structures Query Parameters, section [3.1.4.1.8](#).

pnEnumValues: This parameter must adhere to the parameter specification in PRINTER_ENUM_VALUES Structures Query Parameters, section [3.1.4.1.8](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Perform the validation steps that are specified in PRINTER_ENUM_VALUES Structures Query Parameters, section [3.1.4.1.8](#).

If parameter validation fails, the server MUST fail the operation immediately and return a non-zero error response to the client. [<141>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Enumerate all the values referenced by the specified printer data key.
- Using the enumerated objects, perform the processing and response steps specified in PRINTER_ENUM_VALUES Structures Query Parameters, section [3.1.4.1.8](#).
- Return the status of the operation. [<142>](#)

3.1.4.2.21 RpcEnumPrinterKey (Opnum 80)

RpcEnumPrinterKey enumerates the subkeys of a specified key for a specified printer.

```
DWORD RpcEnumPrinterKey(  
    [in] PRINTER_HANDLE hPrinter,  
    [in, string] const wchar_t* pKeyName,  
    [out, size_is(cbSubkey/sizeof(wchar_t))]  
        wchar_t* pSubkey,  
    [in] DWORD cbSubkey,  
    [out] DWORD* pcbSubkey  
);
```

hPrinter: This parameter MUST specify a handle to a printer object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pKeyName: This parameter MUST be a non-null pointer to a string that MUST specify the key containing the subkeys to enumerate. For rules governing key names, see section [2.2.4.17](#).

pSubkey: This parameter MAY be null if **cbSubkey** equals zero; otherwise, it MUST be a non-null pointer to [BUFFER](#) as specified in String Query Parameters, section [3.1.4.1.6](#).

cbSubkey: This parameter is synonymous with the **cbBuf** parameter as specified in String Query Parameters, section [3.1.4.1.6](#).

pcbSubkey: This parameter is synonymous with the **pcbNeeded** parameter as specified in String Query Parameters, section [3.1.4.1.6](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Perform the validation steps that are specified in String Query Parameters, section [3.1.4.1.6](#), substituting ERROR_MORE_DATA (as specified in [MS-ERREF]) for ERROR_INSUFFICIENT_BUFFER.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<143>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Enumerate the key names that have the key specified in the string that is pointed to by the **pKeyName** parameter as an immediate parent.
- Using the enumerated objects, perform the processing and response steps that are specified in String Query Parameters, section [3.1.4.1.6](#).
- Return the status of the operation. [<144>](#)

3.1.4.2.22 RpcDeletePrinterDataEx (Opnum 81)

RpcDeletePrinterDataEx deletes a specified value from a printer's configuration data, which consists of a set of named and typed values stored in a hierarchy of registry keys.

```
DWORD RpcDeletePrinterDataEx(  
    [in] PRINTER_HANDLE hPrinter,  
    [in, string] const wchar_t* pKeyName,  
    [in, string] const wchar_t* pValueName  
);
```

hPrinter: This parameter MUST specify a handle to a printer object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pKeyName: This parameter MUST be a non-null pointer to a string that MUST specify the key containing the value to delete. For rules governing key names, see section [2.2.4.17](#).

pValueName: This parameter MUST be a non-null pointer to a string that MUST specify the name of the value to delete. For rules governing value names, see section [2.2.4.16](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Additional validation MAY [<145>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately, returning a nonzero error response to the client. [<146>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Delete the printer data value indicated by the **pKeyName** and **pValueName** parameters.
- Return the status of the operation. [<147>](#)

3.1.4.2.23 RpcDeletePrinterKey (Opnum 82)

RpcDeletePrinterKey deletes a specified key and all of its subkeys for a specified printer.

```
DWORD RpcDeletePrinterKey(  
    [in] PRINTER_HANDLE hPrinter,  
    [in, string] const wchar_t* pKeyName  
);
```

hPrinter: This parameter MUST specify a handle to a printer object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pKeyName: This parameter MUST be a non-null pointer to a string that MUST specify the key to delete. For rules governing key names, see section [2.2.4.17](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Additional validation MAY [<148>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<149>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Delete the printer data key indicated by the **pKeyName** parameter and all the subkeys of that key.
- Return the status of the operation. [<150>](#)

3.1.4.2.24 RpcAddPerMachineConnection (Opnum 85)

RpcAddPerMachineConnection adds a remote printer name to the list of supported printer connections for every user who logs onto the server.

```
DWORD RpcAddPerMachineConnection(  
    [in, string, unique] STRING_HANDLE pServer,  
    [in, string] const wchar_t* pPrinterName,  
    [in, string] const wchar_t* pPrintServer,  
    [in, string] const wchar_t* pProvider  
);
```

pServer: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pPrinterName: This parameter MUST adhere to the parameter specification in Printer Name Parameters, section [3.1.4.1.1](#). This parameter MUST specify a printer connection and it MUST be of the form:

SERVER_NAME LOCAL_PRINTER_NAME [with a non-empty SERVER_NAME.]

pPrintServer: This parameter MUST be a non-null pointer to a **string** that MUST specify the name of the server that is hosting the printer to which the connection MUST be established. For rules governing server names, see section [2.2.4.1](#).

pProvider: This parameter MUST be null or else it MUST be a non-null pointer to a **string** that MUST specify the name of the print provider. If the value is null, an implementation-specific default print provider MUST be used. For rules governing print provider names, see section [2.2.4.14.<151>](#)

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Printer Name Parameters, section [3.1.4.1.1](#).
- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Verify that a printer with the same name does not already exist, and if that verification fails, return ERROR_PRINTER_ALREADY_EXISTS, as specified in [\[MS-ERREF\]](#).
- Additional validation MAY [<152>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<153>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Create a printer object that represents the connection.
- Trigger all current users on the machine to create a printer connection to the printer specified in **pPrinterName**.
- Return the status of the operation. [<154>](#)

3.1.4.2.25 RpcDeletePerMachineConnection (Opnum 86)

RpcDeletePerMachineConnection deletes information about a server and connection provider.

```
DWORD RpcDeletePerMachineConnection(  
    [in, string, unique] STRING_HANDLE pServer,  
    [in, string] const wchar_t* pPrinterName  
);
```

pServer: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pPrinterName: This parameter MUST adhere to the parameter specification in Printer Name Parameters, section [3.1.4.1.1](#). This parameter MUST specify a printer connection.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform validation steps as specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform validation steps as specified in Printer Name Parameters, section [3.1.4.1.1](#), [<155>](#)

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<156>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Delete the printer connection that is identified by the string that is pointed to by the **pPrinterName** parameter.
- Return the status of the operation. [<157>](#)

3.1.4.2.26 RpcEnumPerMachineConnections (Opnum 87)

Enumerates each of the connections and copies [PRINTER_INFO_4](#) structures (see section [2.2.1.10.5](#)) for all the per-machine connections into the buffer **pPrinterEnum**.

```
DWORD RpcEnumPerMachineConnections(  
    [in, string, unique] STRING_HANDLE pServer,  
    [in, out, unique, size_is(cbBuf), disable_consistency_check]  
    BYTE* pPrinterEnum,  
    [in] DWORD cbBuf,  
    [out] DWORD* pcbNeeded,  
    [out] DWORD* pcReturned  
);
```

pServer: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pPrinterEnum: This parameter MAY be null if **cbBuf** equals zero; otherwise, it MUST be a non-null pointer to the [BUFFER](#), as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

BUFFER TYPE: `_PRINTER_INFO_4`

cbBuf: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcReturned: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<158>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Enumerate all printers that represent per-machine printer connections on the server that is identified by the **pServer** parameter.
- Using the enumerated objects, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Return the status of the operation. [<159>](#)

3.1.4.2.27 RpcSendRecvBidiData (Opnum 97)

RpcSendRecvBidiData sends and receives bidirectional data. This method is used to communicate with port monitors that support such data.

```
DWORD RpcSendRecvBidiData(  
    [in] PRINTER_HANDLE hPrinter,  
    [in, string, unique] const wchar_t* pAction,  
    [in] RPC_BIDI_REQUEST_CONTAINER* pReqData,  
    [out] RPC_BIDI_RESPONSE_CONTAINER** ppRespData  
);
```

hPrinter: This parameter MUST specify a handle to a printer object or port object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pAction: This parameter MUST be a non-null pointer to a string that MUST specify an action to take. The following actions MUST be supported. Port monitors MAY support additional, implementation-specific action strings. [<160>](#)

Name/Value	Meaning
BIDI_ACTION_ENUM_SCHEMA "EnumSchema"	The method MUST enumerate the supported schemas. The pReqData parameter MUST be ignored. The method MUST store one or more values that correspond to supported schema entries

Name/Value	Meaning
	in the buffer that is pointed to by the ppRespData parameter.
BIDI_ACTION_GET "Get"	The method MUST return the specific value item requested. The pReqData parameter MUST specify a single value entry in the schema. The method MUST store the value of that entry in the buffer that is pointed to by the ppRespData parameter.
BIDI_ACTION_SET "Set"	The method MUST store the supplied data in a single value item in the schema. The pReqData parameter MUST specify a single value entry for the schema and the new value to be stored there. This action MUST NOT change the contents of the buffer that is pointed to by the ppRespData parameter.
BIDI_ACTION_GET_ALL "GetAll"	The method MUST return one or more value items that are reachable from the requested schema item. The pReqData parameter MUST specify an entry in the schema, which is either a value item or an inner schema entry. The action MUST store one or more value entries, and their associated values, in the buffer that is pointed to by the ppRespData parameter.

pReqData: This parameter MUST specify a non-null pointer to an [RPC_BIDI_REQUEST_CONTAINER](#) structure that MUST contain the requested binary data. For details about **RPC_BIDI_REQUEST_CONTAINER**, see section [2.2.1.2.10](#).

ppRespData: This parameter MUST specify a non-null pointer to a variable that MUST receive a pointer to a [RPC_BIDI_RESPONSE_CONTAINER](#) structure containing the response binary data. For details on **RPC_BIDI_RESPONSE_CONTAINER**, see section [2.2.1.2.11](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that the string that is pointed to by the **pAction** parameter specifies a valid command.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<161>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- If the **hPrinter** parameter is a handle to a printer object, load the executable object of the monitor supporting the port associated with the printer identified by **hPrinter**.
- If the **hPrinter** parameter is a handle to a port object, load the executable object of the monitor supporting the port identified by **hPrinter**.
- Invoke the method in that library that is identified by the value of the **pAction** parameter and pass **pReqData** to that method.

- Copy the data that is sent from the action method in the buffer that is pointed to by the **pRespData** parameter; the number of response items MUST match the number of request items.
- Return the status of the operation. <162>

The server MUST NOT change the list of printer objects representing pushed printers as part of processing this method.

3.1.4.3 Job Management Methods

This section specifies methods for discovering, defining, and scheduling print jobs.

Method	Description
RpcSetJob	RpcSetJob pauses, resumes, cancels, or restarts a print job. It also sets print job parameters, such as the job priority and the document name. Opnum 2
RpcGetJob	RpcGetJob retrieves information about a specified print job. Opnum 3
RpcEnumJobs	RpcEnumJobs retrieves information about a specified set of print jobs for a specified printer. Opnum 4
RpcAddJob	RpcAddJob adds a print job to the list of jobs that the print spooler can schedule and retrieves the name of the file used to store the job. Opnum 24
RpcScheduleJob	RpcScheduleJob schedules a print job created by calling RpcAddJob on the print spooler. Opnum 25

3.1.4.3.1 RpcSetJob (Opnum 2)

RpcSetJob pauses, resumes, cancels, or restarts a print job. It also sets print job parameters, such as the job priority and the document name.

```

DWORD RpcSetJob(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD JobId,
    [in, unique] JOB_CONTAINER* pJobContainer,
    [in] DWORD Command
);

```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

JobId: The value of this parameter MUST specify the identifier of the print job. The identifier MUST NOT be zero.

pJobContainer: If the value of the **Command** parameter is zero, this parameter MUST specify a non-null pointer to a [JOB_CONTAINER](#) structure that specifies the parameters that this

method MUST set on the job object. If **pJobContainer** is not null, the **Level** member of the **JOB_CONTAINER** that is pointed to by the **pJobContainer** parameter MUST have a value of one, two, three, or four; and the [JOB_INFO](#) members, **JobId**, **PrinterName**, **ServerName**, **PrinterDriverName**, **Size**, **Submitted**, **Time**, and **TotalPages**, MUST be ignored. For details about **JOB_CONTAINER** structures, see section [2.2.1.2.5](#).

Command: The value of this parameter MUST specify a job control action. It MUST be one of the values listed in the following table. For rules governing job control values, see section [2.2.4.4](#).

Job Control Action Name/Value	Meaning
0x00000000	Perform no additional action.
JOB_CONTROL_PAUSE 0x00000001	Pause the print job.
JOB_CONTROL_RESUME 0x00000002	Resume a paused print job.
JOB_CONTROL_CANCEL 0x00000003	Delete a print job.
JOB_CONTROL_RESTART 0x00000004	Restart a print job.
JOB_CONTROL_DELETE 0x00000005	Delete a print job.
JOB_CONTROL_SENT_TO_PRINTER 0x00000006	Used by port monitors to end a print job; not sent over the wire.
JOB_CONTROL_LAST_PAGE_EJECTED 0x00000007	Used by language monitors to end a print job; not sent over the wire.
JOB_CONTROL_RETAIN 0x00000008	Keep the print job in the printer queue after it prints.
JOB_CONTROL_RELEASE 0x00000009	Release the print job, undoing the effect of JOB_CONTROL_RETAIN .

Return Values:

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that the value of the **JobId** parameter corresponds to a job in the list of jobs.
- Verify that the **pJobContainer** parameter is null, or that it points to a **JOB_CONTAINER** that includes a valid supported level and a non-null pointer to a structure, and that all members of

that structure comply with the constraints defined in section [2.2.1.3.3](#), with the exception of **pMachineName**, which SHOULD be ignored. If that verification fails, return `ERROR_INVALID_PARAMETER`, as specified in [MS-ERREF].

- Verify that the **Command** parameter is a supported command.
- Additional validation MAY [<163>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<164>](#)

Otherwise the server MUST process the message and compose a response to the client as follows:

- Modify the job to reflect the required changes based on the value of **Command**:
 - Pausing the print job (0x00000001): Pause the current job specified by **JobId** and allow any succeeding job to print.
 - Resuming the print job (0x00000002): Resume the job specified by **JobId**.
 - Canceling the print job (0x00000003): Cancel the job specified by **JobId**.
 - Restarting the print job (0x00000004): Reinitialize the internal state of the job specified by **JobId** and re-schedule the job for printing.
 - Deleting the print job (0x00000005): Delete the job specified by **JobId** and any internal structures representing that job.
 - Sent print job to the printer (0x00000006): MAY be set by port monitors associated with a port to signal the job has been sent completely to the device, but it is not sent over the wire.
 - Last Page Ejected (0x00000007): MAY be set by language monitors associated with a port to signal that the physical printer ejected the last page of the job, but it is not sent over the wire.
 - Retain Job (0x00000008): Keep the print job in the printer queue after printing is finished. It MAY then be restarted.
 - Release Job (0x00000009): Release a job previously retained. If the job has finished printing and has not been restarted, it will be removed from the queue.
- If any clients have registered for notification of a job object change, those clients SHOULD be sent notifications about the job changes that the server performed.
- Modify the job that corresponds to the data in the **JOB_CONTAINER** that is pointed to by the **pJobContainer** parameter.
- Return the status of the operation. [<165>](#)

3.1.4.3.2 RpcGetJob (Opnum 3)

RpcGetJob retrieves information about a specified print job.

```
DWORD RpcGetJob(  
    [in] PRINTER_HANDLE hPrinter,  
    [in] DWORD JobId,  
    [in] DWORD Level,  
    [in, out, unique, size_is(cbBuf), disable_consistency_check]  
    BYTE* pJob,
```

```

[in] DWORD cbBuf,
[out] DWORD* pcbNeeded
);

```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

JobId: The value of this parameter MUST specify the identifier of the print job. The identifier MUST NOT be zero.

Level: This value MUST be 0x00000001, 0x00000002, 0x00000003, or 0x00000004.

pJob: This parameter MAY be null if **cbBuf** equals zero; otherwise, it MUST be a non-null pointer to [BUFFER](#) as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

BUFFER TYPE: _JOB_INFO.

cbBuf: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that the value of the **JobId** parameter corresponds to a job in the list of jobs.
- Perform the validation steps that are specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<166>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Using information about the job, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Return the status of the operation. [<167>](#)

3.1.4.3.3 RpcEnumJobs (Opnum 4)

RpcEnumJobs retrieves information about a specified set of print jobs for a specified printer.

```

DWORD RpcEnumJobs (

```

```

[in] PRINTER_HANDLE hPrinter,
[in] DWORD FirstJob,
[in] DWORD NoJobs,
[in] DWORD Level,
[in, out, unique, size_is(cbBuf), disable_consistency_check]
    BYTE* pJob,
[in] DWORD cbBuf,
[out] DWORD* pcbNeeded,
[out] DWORD* pcReturned
);

```

hPrinter: This parameter MUST specify a handle to a printer object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

FirstJob: The value of this parameter MUST specify the zero-based position within the print queue of the first print job to enumerate.

NoJobs: The value of this parameter MUST specify the total number of print jobs to enumerate.

Level: The value of this parameter MUST be 0x00000001, 0x00000002, 0x00000003, or 0x00000004.

pJob: This parameter MAY be null if **cbBuf** equals zero; otherwise, it MUST be a non-null pointer to the [BUFFER](#), as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

BUFFER TYPE: _JOB_INFO.

cbBuf: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcReturned: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Perform the validation steps that are specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<168>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Enumerate as many jobs on the printer as specified by the value of the **NoJobs** parameter (or fewer, if there are fewer jobs), starting with the job whose index is as specified by the **FirstJob** parameter.
- Using the enumerated objects, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Return the status of the operation. [<169>](#)

3.1.4.3.4 RpcAddJob (Opnum 24)

RpcAddJob adds a print job to the list of jobs that the print server can schedule and retrieves the name of the file used to store the job.

```
DWORD RpcAddJob(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check]
    BYTE* pAddJob,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded
);
```

hPrinter: This parameter MUST specify a handle to a printer object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) method.

Level: The value of this parameter MUST specify the type of print job information data structure that the function stores in the buffer that is pointed to by **pAddJob**. The value of this parameter MUST be 0x00000001 or 0x00000002.

pAddJob: This parameter MAY be null if **cbBuf** equals zero; otherwise, it MUST be a non-null pointer to a buffer that MUST contain an [ADDJOB_INFO_1](#) data structure, which is specified in section [2.2.2.3](#).

If the value of the **Level** parameter is two or three, the ADDJOB_INFO_1 structure MUST contain the client machine name in its **PathArray** (located by **PathOffset**).

cbBuf: The value of this parameter MUST specify the size, in bytes, of the buffer to which **pAddJob** points.

pcbNeeded: This parameter MUST be a non-null pointer to a variable that receives the number of bytes copied to the buffer pointed to by **pAddJob**. If the buffer is too small, the variable MUST receive the number of bytes required.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Processing this method is optional. A server MAY return **ERROR_INVALID_PARAMETER** if it does not implement this method. [<170>](#) Otherwise, upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that no other job has been associated with **hPrinter** using [RpcStartDocPrinter](#), and if that verification fails, return **ERROR_INVALID_PARAMETER**, as specified in [MS-ERREF].
- If the value of the **Level** parameter is two or three, verify that the **PathOffset** and **PathArray** members of the ADDJOB_INFO_1 structure pointed to by **pAddJob** are initialized and identify a client name.
- Verify that the value of the **cbBuf** parameter is large enough to hold the ADDJOB_INFO_1 resulting from processing the call, and if that verification fails, return **ERROR_INSUFFICIENT_BUFFER**, as specified in [MS-ERREF].
- Additional validation MAY [<171>](#) be performed.

If parameter validation fails, the server MUST [<172>](#) fail the operation immediately and return a non-zero error response to the client.

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Create a new job object. The server MUST be able to create multiple active job objects on the same printer object.
- If the value of the **Level** parameter is two or three, associate the client name specified in the structure pointed to by **pAddJob** with the new job.
- Store the non-null identifier for the job (see [JOB INFO](#), **JobId** in section [2.2.1.3.3](#)) in the **JobId** member of the structure pointed to by **pAddJob**.
- Store the path name of the file used as spool-file in the **PathArray** member of the structure pointed to by **pAddJob** and initialize the **PathOffset** member to locate **PathArray**.
- Store the new size of the ADDJOB_INFO_1 structure in the variable that is pointed to by **pcbNeeded**.
- If any clients are registered for notification of the job object change, a notification MUST be broadcast to them.
- Return the status [<173>](#) of the operation.

3.1.4.3.5 RpcScheduleJob (Opnum 25)

RpcScheduleJob schedules a print job created by calling [RpcAddJob](#) on the print server.

```
DWORD RpcScheduleJob(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD JobId
);
```

hPrinter: This parameter MUST specify a handle to a printer object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

JobId: The value of this parameter MUST specify the identifier of the print job.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that no other job was added by using [RpcStartDocPrinter](#) and is pending on the printer object that is identified by **hPrinter**. If that verification fails, return ERROR_SPL_NO_ADDJOB, as specified in [MS-ERREF].
- Verify that **JobId** identifies an existing job, and if that verification fails, return [ERROR_INVALID_PARAMETER](#).
- Verify that the job has not already been scheduled, and if that verification fails, return ERROR_INVALID_PARAMETER.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<174>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Schedule the printing job that is identified by **JobId**.
- If any clients are registered for notification of the job object change, a notification MUST be broadcast to them.
- Return the status [<175>](#) of the operation.

3.1.4.4 Printer Driver Management Methods

This section specifies methods for discovering and installing printer drivers.

Method	Description
RpcAddPrinterDriver	RpcAddPrinterDriver installs a printer driver on the print server and links the configuration, data, and printer driver files. Opnum 9
RpcEnumPrinterDrivers	RpcEnumPrinterDrivers enumerates the printer drivers installed on a specified print server. Opnum 10
RpcGetPrinterDriver	RpcGetPrinterDriver retrieves printer driver data for the specified printer. Opnum 11
RpcGetPrinterDriverDirectory	RpcGetPrinterDriverDirectory retrieves the path of the printer driver directory. Opnum 12
RpcDeletePrinterDriver	RpcDeletePrinterDriver removes the specified printer driver from

Method	Description
	the list of supported drivers for a server. Opnum 13
RpcGetPrinterDriver2	RpcGetPrinterDriver2 retrieves printer driver data for the specified printer. Opnum 53
RpcDeletePrinterDriverEx	RpcDeletePrinterDriverEx removes the specified printer driver from the list of supported drivers for a server and deletes the files associated with the printer driver. This method also can delete specific versions of the printer driver. Opnum 84
RpcAddPrinterDriverEx	RpcAddPrinterDriverEx installs a printer driver on the print server. This method performs a similar function as RpcAddPrinterDriver (see section 3.1.4.4.1) and is also used to specify options that permit strict upgrade, strict downgrade, copying of newer files only, and copying of all files (regardless of their time stamps). Opnum 89

3.1.4.4.1 RpcAddPrinterDriver (Opnum 9)

RpcAddPrinterDriver installs a printer driver on the print server and links the configuration, data, and printer driver files.

```
DWORD RpcAddPrinterDriver(
    [in, string, unique] STRING_HANDLE pName,
    [in] DRIVER_CONTAINER* pDriverContainer
);
```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pDriverContainer: This parameter MUST adhere to the parameter specification in DRIVER_CONTAINER Parameters, section [3.1.4.1.9.3](#). The **Level** member of the **DRIVER_CONTAINER** MUST be two, three, or four.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform validation steps as specified in DRIVER_CONTAINER Parameters, section [3.1.4.1.9.3](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<176>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Copy the printer driver files to their destination. If the copy operation fails, the server MUST fail the call immediately and return a nonzero error response to the client.[.<177>](#)
- Create the printer driver object.
- If any clients have registered for notification of server object changes, a notification MUST be broadcast to them.
- Return the status of the operation.[.<178>](#)

3.1.4.4.2 RpcEnumPrinterDrivers (Opnum 10)

RpcEnumPrinterDrivers enumerates the printer drivers installed on a specified print server.

```
DWORD RpcEnumPrinterDrivers(  
    [in, string, unique] STRING_HANDLE pName,  
    [in, string, unique] wchar_t* pEnvironment,  
    [in] DWORD Level,  
    [in, out, unique, size_is(cbBuf), disable_consistency_check]  
    BYTE* pDrivers,  
    [in] DWORD cbBuf,  
    [out] DWORD* pcbNeeded,  
    [out] DWORD* pcReturned  
);
```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pEnvironment: This parameter MUST adhere to the parameter specification in Environment Name Parameters, section [3.1.4.1.4.<179>](#)

Level: This value refers to the level of driver information structure, as follows:

Value	Meaning
0x00000001	Corresponds to _DRIVER_INFO_1, specified in section 2.2.2.5.1 .
0x00000002	Corresponds to _DRIVER_INFO_2, specified in section 2.2.2.5.2 .
0x00000003	Corresponds to _DRIVER_INFO_3, specified in section 2.2.2.5.3 .
0x00000004	Corresponds to _DRIVER_INFO_4, specified in section 2.2.2.5.4 .
0x00000005	Corresponds to _DRIVER_INFO_5, specified in section 2.2.2.5.5 .
0x00000006	Corresponds to _DRIVER_INFO_6, specified in section 2.2.2.5.6 .
0x00000008	Corresponds to _DRIVER_INFO_8, specified in section 2.2.2.5.8 .

pDrivers: This parameter MAY be null if **cbBuf** equals zero; otherwise, it MUST be a non-null pointer to the [BUFFER](#), as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

BUFFER TYPE: _DRIVER_INFO.

cbBuf: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcReturned: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in Environment Name Parameters, section [3.1.4.1.4](#).
- Perform the validation steps that are specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<180>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Enumerate all drivers on the specified print server that match the requested environment.
- Using the enumerated objects, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Return the status of the operation. [<181>](#)

3.1.4.4.3 RpcGetPrinterDriver (Opnum 11)

RpcGetPrinterDriver retrieves printer driver data for the specified printer.

```
DWORD RpcGetPrinterDriver(  
    [in] PRINTER_HANDLE hPrinter,  
    [in, string, unique] wchar_t* pEnvironment,  
    [in] DWORD Level,  
    [in, out, unique, size_is(cbBuf), disable_consistency_check]  
    BYTE* pDriver,  
    [in] DWORD cbBuf,  
    [out] DWORD* pcbNeeded  
);
```

hPrinter: This parameter MUST specify a handle to a printer object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pEnvironment: This parameter MUST adhere to the parameter specification in Environment Name Parameters, section [3.1.4.1.4](#).

Level: This value refers to the level of driver information structure, as follows:

Value	Meaning
0x00000001	Corresponds to _DRIVER_INFO_1, specified in section 2.2.2.5.1
0x00000002	Corresponds to _DRIVER_INFO_2, specified in section 2.2.2.5.2
0x00000003	Corresponds to _DRIVER_INFO_3, specified in section 2.2.2.5.3
0x00000004	Corresponds to _DRIVER_INFO_4, specified in section 2.2.2.5.4
0x00000005	Corresponds to _DRIVER_INFO_5, specified in section 2.2.2.5.5
0x00000006	Corresponds to _DRIVER_INFO_6, specified in section 2.2.2.5.6
0x00000008	Corresponds to _DRIVER_INFO_8, specified in section 2.2.2.5.8

pDriver: This parameter MAY be null if **cbBuf** equals zero; otherwise, it MUST be a non-null pointer to [BUFFER](#) as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

BUFFER TYPE: _DRIVER_INFO.

cbBuf: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Perform the validation steps that are specified in Environment Name Parameters, section [3.1.4.1.4](#).
- Perform the validation steps that are specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<182>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Using the information about the printer driver, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Return the status of the operation. [<183>](#)

3.1.4.4.4 RpcGetPrinterDriverDirectory (Opnum 12)

RpcGetPrinterDriverDirectory retrieves the path of the printer driver directory.

```
DWORD RpcGetPrinterDriverDirectory(  
    [in, string, unique] STRING_HANDLE pName,  
    [in, string, unique] wchar_t* pEnvironment,  
    [in] DWORD Level,  
    [in, out, unique, size_is(cbBuf), disable_consistency_check]  
    BYTE* pDriverDirectory,  
    [in] DWORD cbBuf,  
    [out] DWORD* pcbNeeded  
);
```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pEnvironment: This parameter MUST adhere to the parameter specification in Environment Name Parameters, section [3.1.4.1.4](#).

pDriverDirectory: This parameter MAY be null if **cbBuf** equals zero; otherwise, it MUST be a non-null pointer to [BUFFER](#), as specified in String Query Parameters, section [3.1.4.1.6](#).

cbBuf: This parameter MUST adhere to the parameter specification in String Query Parameters, section [3.1.4.1.6](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in String Query Parameters, section [3.1.4.1.6](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in Environment Name Parameters, section [3.1.4.1.4](#).
- Perform the validation steps that are specified in String Query Parameters, section [3.1.4.1.6](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<184>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- With the path of the printer driver directory on the print server, perform the processing and response steps specified in String Query Parameters, section [3.1.4.1.6](#).
- Return the status of the operation. [<185>](#)

3.1.4.4.5 RpcDeletePrinterDriver (Opnum 13)

RpcDeletePrinterDriver removes the specified printer driver from the list of supported drivers for a server.

```
DWORD RpcDeletePrinterDriver(  
    [in, string, unique] STRING_HANDLE pName,  
    [in, string] wchar_t* pEnvironment,  
    [in, string] wchar_t* pDriverName  
);
```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pEnvironment: This parameter MUST adhere to the parameter specification in Environment Name Parameters, section [3.1.4.1.4](#).

pDriverName: This parameter MUST be a non-null pointer to a string that MUST specify the name of the printer drive to delete. For rules governing printer driver names, see section [2.2.4.6](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in Environment Name Parameters, section [3.1.4.1.4](#).
- Verify that the string pointed to by the **pDriverName** parameter contains the name of a driver that is part of the list of drivers that are installed on the server for the environment specified by the string pointed to by the **pEnvironment** parameter; if that verification fails, return ERROR_UNKNOWN_PRINTER_DRIVER.
- Verify that the printer driver is not used by any printer in the system, and if that verification fails, return ERROR_PRINTER_DRIVER_IN_USE.
- Additional validation MAY [<186>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<187>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Clear all references to this printer driver from any other data structures.
- Delete the printer driver object.
- If any clients have registered for notifications of the server object change, a notification MUST be broadcast to them.

- Return the status of the operation. [<188>](#)

3.1.4.4.6 RpcGetPrinterDriver2 (Opnum 53)

RpcGetPrinterDriver2 retrieves printer driver data for the specified printer.

```
DWORD RpcGetPrinterDriver2(
    [in] PRINTER_HANDLE hPrinter,
    [in, string, unique] wchar_t* pEnvironment,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check]
    BYTE* pDriver,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded,
    [in] DWORD dwClientMajorVersion,
    [in] DWORD dwClientMinorVersion,
    [out] DWORD* pdwServerMaxVersion,
    [out] DWORD* pdwServerMinVersion
);
```

hPrinter: This parameter MUST specify a handle to a printer object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pEnvironment: This parameter MUST adhere to the parameter specification in Environment Name Parameters, section [3.1.4.1.4](#).

Level: This value refers to the level of driver information structure, as follows:

Value	Meaning
0x00000001	Corresponds to _DRIVER_INFO_1, specified in section 2.2.2.5.1 .
0x00000002	Corresponds to _DRIVER_INFO_2, specified in section 2.2.2.5.2 .
0x00000003	Corresponds to _DRIVER_INFO_3, specified in section 2.2.2.5.3 .
0x00000004	Corresponds to _DRIVER_INFO_4, specified in section 2.2.2.5.4 .
0x00000006	Corresponds to _DRIVER_INFO_6, specified in section 2.2.2.5.6 .
0x00000008	Corresponds to _DRIVER_INFO_8, specified in section 2.2.2.5.8 .
0x00000065	Corresponds to _DRIVER_INFO_101, specified in section 2.2.2.5.9 .

pDriver: This parameter MAY be null if **cbBuf** equals zero; otherwise, it MUST be a non-null pointer to the [BUFFER](#), as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

BUFFER TYPE: _DRIVER_INFO.

cbBuf: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

dwClientMajorVersion: The value of this parameter MUST be the implementation-specific major printer driver version of the client operating system.<189>

dwClientMinorVersion: The value of this parameter MUST be the implementation-specific minor printer driver version of the client operating system.<190>

pdwServerMaxVersion: This parameter MUST be null or else it MUST be a non-null pointer to a DWORD that SHOULD receive the implementation-specific major version that the operating system supports for that printer driver.

pdwServerMinVersion: This parameter MUST be null or else it MUST be a non-null pointer to a DWORD that SHOULD receive the implementation-specific minimum version that the operating system supports for that printer driver.<191>

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Perform the validation steps that are specified in Environment Name Parameters, section [3.1.4.1.4](#).
- Perform the validation steps that are specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client.<192>

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Find a printer driver that is compatible with the OS version on the specified print server that is specified by the value of the **dwClientMajorVersion** and **dwClientMinorVersion** parameters.
- Using the identified printer driver, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

If parameter validation was successful, the server SHOULD also perform the following steps:

- Store the actual compatible operating system version<193> for the printer driver in the variables pointed to by the **pdwServerMaxVersion** and **pdwServerMinVersion** parameters.
- Return the status of the operation.<194>

3.1.4.4.7 RpcDeletePrinterDriverEx (Opnum 84)

Removes the specified printer driver from the list of supported drivers for a server and deletes the files associated with the printer driver. This method also can delete specific versions of the printer driver.

```
DWORD RpcDeletePrinterDriverEx(
```

```

[in, string, unique] STRING_HANDLE pName,
[in, string] wchar_t* pEnvironment,
[in, string] wchar_t* pDriverName,
[in] DWORD dwDeleteFlag,
[in] DWORD dwVersionNum
);

```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pEnvironment: This parameter MUST adhere to the parameter specification in Environment Name Parameters, section [3.1.4.1.4](#).

pDriverName: This parameter MUST be a non-null pointer to a string that MUST specify the name of the printer driver to delete. For rules governing printer driver names, see section [2.2.4.6](#).

dwDeleteFlag: The value of this parameter MUST specify the options to apply when deleting files and versions of the printer driver. The value of this parameter MUST be zero or the result of the bitwise OR of zero or more of the following values. If the value of this parameter is zero, the driver MUST be removed from the list of supported drivers and the driver files MUST remain on the print server.

Note All other bits MUST be zero.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	0	C	B	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Where the bits are defined as:

Value	Description
A DPD_DELETE_UNUSED_FILES	Remove unused printer driver files. The method MUST NOT fail if some of the printer driver's files are in use.
B DPD_DELETE_SPECIFIC_VERSION	Delete the version specified by the value of the dwVersionNum parameter. Because more than one version of a printer driver can be installed on a print server, setting this flag does not guarantee that the printer driver will be removed from the list of supported drivers on the print server.
C DPD_DELETE_ALL_FILES	Delete the printer driver only if all its associated files can be removed. The delete operation MUST fail if any of the printer driver's files are in use by another installed printer driver.

dwVersionNum: The value of this parameter MUST specify the version of the printer driver to delete. The value of this parameter is implementation-specific. The value of this parameter identifies the operating system for which the printer driver was written. It has the same format and meaning as the **cVersion** member for the [RPC_DRIVER_INFO](#) structures. This parameter MUST be ignored if the **DPD_DELETE_SPECIFIC_VERSION** flag in the **dwDeleteFlag** parameter is not set. [<195>](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate the following

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in Environment Name Parameters, section [3.1.4.1.4](#).
- Verify that the string pointed to by the **pDriverName** parameter contains the name of a driver that is part of the list of drivers that are installed on the server for the environment specified by the string pointed to by the **pEnvironment** parameter; if that verification fails, return **ERROR_UNKNOWN_PRINTER_DRIVER**.
- Verify that the printer driver identified by **pDriverName** is not being used by any printer in the system; if that verification fails, return **ERROR_PRINTER_DRIVER_IN_USE**.
- Verify that the value of the **dwDeleteFlag** parameter contains the result of the bitwise OR of zero or more of the **DPD_DELETE** defined constants and that all other bits are clear (zero), and if that verification fails, return **ERROR_INVALID_PARAMETER**.
- If the **DPD_DELETE_SPECIFIC_VERSION** bit is set in the **dwDeleteFlag** parameter, verify that the value of the **dwVersionNum** parameter matches the printer driver; if that verification fails, return **ERROR_UNKNOWN_PRINTER_DRIVER**.
- Additional validation MAY [<196>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<197>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Clear references to this version of the printer driver in any other data structures.
- Delete the printer driver object and any associated driver files in compliance with the settings in **dwDeleteFlag**.
- If any clients have registered for notification of server object changes, a notification MUST be broadcast to them.
- Return the status of the operation. [<198>](#)

3.1.4.4.8 RpcAddPrinterDriverEx (Opnum 89)

RpcAddPrinterDriverEx installs a printer driver on the print server. This method performs a similar function as [RpcAddPrinterDriver](#) (see section [3.1.4.4.1](#)) and is also used to specify options that permit strict upgrade, strict downgrade, copying of newer files only, and copying of all files (regardless of their time stamps).

```
DWORD RpcAddPrinterDriverEx(
    [in, string, unique] STRING_HANDLE pName,
```

```

[in] DRIVER_CONTAINER* pDriverContainer,
[in] DWORD dwFileCopyFlags
);

```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pDriverContainer: This parameter MUST adhere to the parameter specification in DRIVER_CONTAINER Parameters, section [3.1.4.1.9.3](#). The **Level** member of the **DRIVER_CONTAINER** MUST be two, three, four, six, or eight; it refers to the level of driver information structure, as follows:

Value	Meaning
2	Corresponds to _DRIVER_INFO_2, specified in section 2.2.2.5.2 .
3	Corresponds to _DRIVER_INFO_3, specified in section 2.2.2.5.3 .
4	Corresponds to _DRIVER_INFO_4, specified in section 2.2.2.5.4 .
6	Corresponds to _DRIVER_INFO_6, specified in section 2.2.2.5.6 .
8	Corresponds to _DRIVER_INFO_8, specified in section 2.2.2.5.8 .

dwFileCopyFlags: The value of this parameter MUST specify the options to use when copying the replacement printer driver files. The value of this parameter MUST be a bitwise OR operator of exactly one of the values from the first table and zero or more of the values in the second table.

Exactly one of the values from the following first table MUST be set:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	D	C	B	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Where the bits are defined as:

Value	Description
A APD_STRICT_UPGRADE	Add the replacement printer driver only if none of the files of the replacement printer driver are older than any corresponding files of the currently installed printer driver.
B APD_STRICT_DOWNGRADE	Add the replacement printer driver only if none of the files of the currently installed printer driver are older than any corresponding files of the replacement printer driver.
C APD_COPY_ALL_FILES	Add the printer driver and copy all the files in the printer driver directory. The file time stamps MUST be ignored with this option.
D APD_COPY_NEW_FILES	Add the printer driver and MUST copy the files in the printer driver directory that are newer than any corresponding files that are

Value	Description
	currently in use.

A bitwise OR of zero or more of the following values MAY additionally be specified:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	E	0	0	0	0	0	0	G	F	0	0	0	0	0	0	0	0	0	0	0	H	0	0	0	0	0	0	0	0

Where the bits are defined as:

Value	Description
E APD_COPY_FROM_DIRECTORY	Add the printer driver by using the fully qualified file names that are specified in the _DRIVER_INFO_6 structure. This flag MUST be set together with one of the other copy flags.
F APD_DONT_COPY_FILES_TO_CLUSTER	The Windows implementation is found in Appendix B: Windows Behavior. <199>
G APD_COPY_TO_ALL_SPOOLERS	Windows implementation is found in Appendix B: Windows Behavior. <200>
H APD_RETURN_BLOCKING_STATUS_CODE	Windows implementation is found in Appendix B: Windows Behavior. <201>

Return Values:

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in DRIVER_CONTAINER Parameters, section [3.1.4.1.9.3](#).
- Verify that **dwFileCopyFlags** contains a valid flag value or set of values as described in the **dwFileCopyFlags** parameter definition above.
- Additional validation MAY [<202>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<203>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Copy the printer driver files to their destinations, in compliance with the settings described by the value of the **dwFileCopyFlags** parameter.
- Create the printer driver object.
- If any clients have registered for notification of server object changes, a notification **MUST** be broadcast to them.
- Return the status of the operation. [<204>](#)

3.1.4.5 Form Management Methods

This section specifies methods for discovering and configuring printer forms.

Method	Description
RpcAddForm	RpcAddForm adds a form name to the list of supported forms. Opnum 30
RpcDeleteForm	RpcDeleteForm removes a form name from the list of supported forms. Opnum 31
RpcGetForm	RpcGetForm retrieves information about a specified form. Opnum 32
RpcSetForm	RpcSetForm replaces the form information for the specified form. Opnum 33
RpcEnumForms	The RpcEnumForms method enumerates the forms that the specified printer supports. Opnum 34

3.1.4.5.1 RpcAddForm (Opnum 30)

RpcAddForm adds a form name to the list of supported forms.

```
DWORD RpcAddForm(
    [in] PRINTER_HANDLE hPrinter,
    [in] FORM_CONTAINER* pFormInfoContainer
);
```

hPrinter: This parameter **MUST** specify a handle to a printer object or server object that **MUST** have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pFormInfoContainer: This parameter **MUST** adhere to the parameter specification in FORM_CONTAINER Parameters, section [3.1.4.1.9.4](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate the following

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Perform the validation steps that are specified in FORM_CONTAINER Parameters, section [3.1.4.1.9.4](#).
- Verify that the form does not already exist, and if that verification fails, return ERROR_FILE_EXISTS, as specified in [MS-ERREF].
- Additional validation MAY [<205>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<206>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Create the form object.
- If any clients have registered for notification of server object changes, a notification MUST be broadcast to them.
- Return the status of the operation. [<207>](#)

3.1.4.5.2 RpcDeleteForm (Opnum 31)

RpcDeleteForm removes a form name from the list of supported forms.

```
DWORD RpcDeleteForm(  
    [in] PRINTER_HANDLE hPrinter,  
    [in, string] wchar_t* pFormName  
);
```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pFormName: This parameter MUST be a non-null pointer to a string that MUST identify the form to delete. For rules governing form names, see section [2.2.4.11](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate the following:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that the **pFormName** parameter points to a string.
- Additional validation MAY [<208>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<209>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Clear the references to this form from any other data structures.
- Delete the form object.
- If any clients have registered for notification of server object changes, a notification MUST be broadcast to them.
- Return the status of the operation. [<210>](#)

3.1.4.5.3 RpcGetForm (Opnum 32)

RpcGetForm retrieves information about a specified form.

```
DWORD RpcGetForm(  
    [in] PRINTER_HANDLE hPrinter,  
    [in, string] wchar_t* pFormName,  
    [in] DWORD Level,  
    [in, out, unique, size_is(cbBuf), disable_consistency_check]  
    BYTE* pForm,  
    [in] DWORD cbBuf,  
    [out] DWORD* pcbNeeded  
);
```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pFormName: This parameter MUST be a non-null pointer to a string that MUST specify the form name for which data is required. For rules governing form names, see section [2.2.4.11](#).

Level: This value refers to the level of form information structure, as follows:

Value	Meaning
0x00000001	Corresponds to _FORM_INFO_1, specified in section 2.2.2.6.1 .
0x00000002	Corresponds to _FORM_INFO_2, specified in section 2.2.2.6.2 .

pForm: This parameter MAY be null if cbBuf equals zero; otherwise, it MUST be a non-null pointer to the [BUFFER](#), as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

BUFFER TYPE: _FORM_INFO.

cbBuf: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that the **pFormName** parameter points to a string.
- Perform the validation steps that are specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<211>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Using information about the form, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Return the status of the operation. [<212>](#)

3.1.4.5.4 RpcSetForm (Opnum 33)

RpcSetForm replaces the form information for the specified form.

```
DWORD RpcSetForm(
    [in] PRINTER_HANDLE hPrinter,
    [in, string] wchar_t* pFormName,
    [in] FORM_CONTAINER* pFormInfoContainer
);
```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pFormName: This parameter MUST be a non-null pointer to a string that MUST specify the form name on which the form information is set. For rules governing form names, see section [2.2.4.11](#).

pFormInfoContainer: This parameter MUST adhere to the parameter specification in FORM_CONTAINER Parameters, section [3.1.4.1.9.4](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Perform the validation steps that are specified in FORM_CONTAINER Parameters, section [3.1.4.1.9.4](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<213>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Modify the object that corresponds to **pFormName** in order to reflect the new settings supplied in **pFormInfoContainer**.
- If any clients have registered for notification of server object changes, a notification MUST be broadcast to them.
- Return the status of the operation. [<214>](#)

3.1.4.5.5 RpcEnumForms (Opnum 34)

The **RpcEnumForms** method enumerates the forms that the specified printer supports.

```
DWORD RpcEnumForms(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check]
    BYTE* pForm,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded,
    [out] DWORD* pcReturned
);
```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

Level: This value refers to the level of form information structure, as follows:

Value	Meaning
0x00000001	Corresponds to _FORM_INFO_1, specified in section 2.2.2.6.1 .
0x00000002	Corresponds to _FORM_INFO_2, specified in section 2.2.2.6.2 .

pForm: This parameter MAY be null if cbBuf equals zero; otherwise, it MUST be a non-null pointer to the [BUFFER](#), as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

BUFFER TYPE: _FORM_INFO.

cbBuf: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcReturned: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Perform the validation steps that are specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Additional validation MAY [<215>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<216>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Enumerate all forms on the printer or print server.
- Using the enumerated objects, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Return the status of the operation. [<217>](#)

3.1.4.6 Port Management Methods

This section specifies methods for discovering and communicating with printer ports.

Method	Description
RpcEnumPorts	RpcEnumPorts enumerates the ports that are available for printing on a specified server.
RpcDeletePort	Removes a port added by the RpcAddPortEx method (see section 3.1.4.6.3).
RpcAddPortEx	RpcAddPortEx adds a port name to the list of supported ports.
RpcSetPort	RpcSetPort sets the status associated with a printer port.
RpcXcvData	RpcXcvData provides an extensible mechanism by which a client can control ports on the server and exchange port-specific commands and data with the server. For more information about language monitor methods, see section 3.1.4.11 .

3.1.4.6.1 RpcEnumPorts (Opnum 35)

RpcEnumPorts enumerates the ports that are available for printing on a specified server.

```
DWORD RpcEnumPorts(  
    [in, string, unique] STRING_HANDLE pName,
```

```

[in] DWORD Level,
[in, out, unique, size_is(cbBuf), disable_consistency_check]
    BYTE* pPort,
[in] DWORD cbBuf,
[out] DWORD* pcbNeeded,
[out] DWORD* pcReturned
);

```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

Level: This value refers to the level of port information structure, as follows:

Value	Meaning
0x00000001	Corresponds to _PORT_INFO_1, specified in section 2.2.2.9.1 .
0x00000002	Corresponds to _PORT_INFO_2, specified in section 2.2.2.6.2 .

pPort: This parameter MAY be null if cbBuf equals zero; otherwise, it MUST be a non-null pointer to the [BUFFER](#), as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

BUFFER TYPE: _PORT_INFO.

cbBuf: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcReturned: This parameter MUST adhere to the parameter in INFO Structures Query Parameters, section [3.1.4.1.5](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<218>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Enumerate all ports on the print server.

- Using the enumerated objects, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Return the status of the operation. [<219>](#)

3.1.4.6.2 RpcDeletePort (Opnum 39)

Removes a port.

```
DWORD RpcDeletePort(
    [in, string, unique] STRING_HANDLE pName,
    [in] ULONG_PTR hWnd,
    [in, string] wchar_t* pPortName
);
```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

hWnd: The value of this parameter SHOULD be zero and MUST be ignored upon receipt.

pPortName: This parameter MUST be a non-null pointer to a string that MUST specify the name of the port that MUST be deleted. For rules governing port names, see section [2.2.4.13](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate the following:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Verify that the string that is referenced by **pPortName** represents a port installed on the server.
- Verify that the port is not being used by any other printer in the system. [<220>](#)

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<221>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Clear the references to this port from any other data structures.
- Delete the port object.
- Modify the list of ports in the system to exclude the deleted port.
- If any clients have registered for notification of server object changes, a notification MUST be broadcast to them.
- Return the status of the operation. [<222>](#)

3.1.4.6.3 RpcAddPortEx (Opnum 61)

RpcAddPortEx adds a port name to the list of supported ports.

```
DWORD RpcAddPortEx(  
    [in, string, unique] STRING_HANDLE pName,  
    [in] PORT_CONTAINER* pPortContainer,  
    [in] PORT_VAR_CONTAINER* pPortVarContainer,  
    [in, string] wchar_t* pMonitorName  
);
```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pPortContainer: This parameter MUST adhere to the parameter specification in PORT_CONTAINER Parameters, section [3.1.4.1.9.5](#). The value of the **Level** member in the [PORT_CONTAINER](#) that is referenced by this parameter MUST be one, two, or 0xFFFFFFFF.

pPortVarContainer: This parameter MUST be a non-null pointer to the [PORT_VAR_CONTAINER](#) information structure that MUST contain information about the port. For details on PORT_VAR_CONTAINER, see section [2.2.1.2.8](#).

pMonitorName: This parameter MUST be a non-null pointer to a string that MUST specify the monitor associated with the port. For rules governing monitor names, see section [2.2.4.12](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in PORT_CONTAINER Parameters, section [3.1.4.1.9.5](#).
- If the value of the **Level** member of the **PORT_CONTAINER** that is referenced by the **pPortContainer** parameter is 0xFFFFFFFF, verify that the **pPortVarContainer** parameter is not null; if that verification fails, return ERROR_INVALID_PARAMETER, as specified in [\[MS-ERREF\]](#).
- Verify that the port does not already exist, and if that verification fails, [<223>](#) return ERROR_ALREADY_EXISTS, as specified in [\[MS-ERREF\]](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<224>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Create a new port that is compatible with the port monitor identified by the string that is referenced by the **pMonitorName** parameter; if the **Level** member of the **PORT_CONTAINER** is 0xFFFFFFFF, pass the data that is contained in the **PORT_VAR_CONTAINER** that is pointed to by the **pPortVarContainer** parameter to the port monitor.

- Associate the new port with the port monitor identified by the string that is referenced by the **pMonitorName** parameter.
- Modify the list of ports in the system to include the port created by this method.
- If any clients have registered for notification of server object changes, a notification **MUST** be broadcast to them.
- Return the status of the operation. [<225>](#)

3.1.4.6.4 RpcSetPort (Opnum 71)

RpcSetPort sets the status associated with a printer port.

```
DWORD RpcSetPort(
    [in, string, unique] STRING_HANDLE pName,
    [in, string, unique] wchar_t* pPortName,
    [in] PORT_CONTAINER* pPortContainer
);
```

pName: This parameter **MUST** adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pPortName: This parameter **MUST** be a non-null pointer to a string that **MUST** specify the name of the printer port. For rules governing port names, see section [2.2.4.13](#).

pPortContainer: This parameter **MUST** adhere to the parameter specification in PORT_CONTAINER Parameters, section [3.1.4.1.9.5](#). The level as specified in the **Level** member of the [PORT_CONTAINER](#) structure **MUST** be three.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server **MUST** validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in PORT_CONTAINER Parameters, section [3.1.4.1.9.5](#).
- Additional validation **MAY** [<226>](#) be performed.

If parameter validation fails, the server **MUST** fail the operation immediately and return a nonzero error response to the client. [<227>](#)

Otherwise, the server **MUST** process the message and compose a response to the client as follows:

- Copy the **dwStatus**, **pszStatus**, and **dwSeverity** members from [PORT_INFO](#) that is referenced by the **PORT_CONTAINER** pointed to by the **pPortContainer** parameter to the port object that is specified by the string that is reference by the **pPortName** parameter.
- Return the status of the operation. [<228>](#)

3.1.4.6.5 RpcXcvData (Opnum 88)

RpcXcvData provides an extensible mechanism by which a client can control ports on the server and exchange port specific commands and data with the server. [<229>](#)

```
DWORD RpcXcvData(  
    [in] PRINTER_HANDLE hXcv,  
    [in, string] const wchar_t* pszDataName,  
    [in, size_is(cbInputData)] BYTE* pInputData,  
    [in] DWORD cbInputData,  
    [out, size_is(cbOutputData)] BYTE* pOutputData,  
    [in] DWORD cbOutputData,  
    [out] DWORD* pcbOutputNeeded,  
    [in, out] DWORD* pdwStatus  
);
```

hXcv: This parameter MUST specify a handle to a port object, which MUST have been opened using the [RpcOpenPrinter](#) or [RpcOpenPrinterEx](#) methods.

pszDataName: This parameter MUST be a non-null pointer to a string representing the name of the requested data or action. The following table shows the actions that SHOULD be supported. Other port monitor-specific action strings MAY be supported. [<230>](#)

Value	Meaning
AddPort	Add an instance of a specific port type controlled by the port monitor.
DeletePort	Delete an instance of a specific port type controlled by the port monitor.
MonitorUI	The action SHOULD return the name of the associated port monitor client-side executable configuration module in the buffer that is referenced by the pOutputData parameter.

pInputData: This parameter MAY be null if **cbInputData** equals zero; otherwise, it MUST be a non-null pointer to a buffer that MUST contain input data.

cbInputData: The value of this parameter MUST contain the size, in bytes, of the buffer that is pointed to by the **pInputData** parameter.

pOutputData: This parameter MAY be null if **cbOutputData** equals zero; otherwise, it MUST contain a non-null pointer to a buffer to receive output data.

cbOutputData: The value of this parameter MUST contain the size, in bytes, of the buffer that is pointed to by the **pOutputData** parameter.

pcbOutputNeeded: This parameter MUST be a non-null pointer to a location that MUST receive the size, in bytes, required for the buffer that is pointed to by the **pOutputData** parameter.

pdwStatus: This parameter MUST be a non-null pointer to a variable that MUST receive the status value that is returned by the port monitor's **XcvData** method. The value stored is zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate that the print server successfully called the port monitor's XcvData method, or a non-zero Windows error code to indicate failure, as specified in [MS-ERREF].

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#), substituting **hXcv** for **hPrinter**.
- Verify that the string that is referenced by the **pszDataName** parameter is a valid command name.
- Verify that the **pInputData** parameter is null or, if it is not null, that the value of the **cbInputData** parameter is not zero.
- Verify that the **pOutputData** parameter is null; if it is not null and the action returns information in **pOutputData**, verify that the value of the **cbOutputData** parameter is not zero.
- For actions that return data in the buffer that is pointed to by the **pOutputData** parameter, verify that the size of the buffer, as specified by the value of the **cbOutputData** parameter, is sufficient. If that verification fails, store the required buffer size in the variable that is pointed to by the **pcbOutputNeeded** parameter and return ERROR_INSUFFICIENT_BUFFER, as specified in [MS-ERREF].

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<231>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Load the executable object of the monitor that supports the port that is associated with the port object that is referenced by the **hXcv** parameter.
- Invoke the method in that library that is identified by the string that is pointed to by the **pszDataName** parameter, and pass in the **pInputData** parameter.
- If the **pOutputData** parameter is not null, copy the output data from the method into **pOutputData**, up to the limit that is specified by the value of the **cbOutputData** parameter.
- Write the size of the data that was copied into **pcbOutputNeeded**.
- If the **pdwStatus** parameter is not null, store the status of the port in the variable that is referenced by the **pdwStatus** parameter.
- Return the status of the operation. [<232>](#)

3.1.4.7 Port Monitor Management Methods

This section specifies methods for discovering and installing port monitors.

Method	Description
RpcEnumMonitors	The RpcEnumMonitors method retrieves information about the port monitors installed on the specified server.

Method	Description
	Opnum 36
RpcAddMonitor	RpcAddMonitor installs a local port monitor and links the configuration, data, and monitor files. Opnum 46
RpcDeleteMonitor	RpcDeleteMonitor removes a port monitor. Opnum 47

3.1.4.7.1 RpcEnumMonitors (Opnum 36)

The **RpcEnumMonitors** method retrieves information about the port monitors installed on the specified server.

```

DWORD RpcEnumMonitors(
    [in, string, unique] STRING_HANDLE pName,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check]
    BYTE* pMonitor,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded,
    [out] DWORD* pcReturned
);

```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

Level: This value refers to the level of port monitor information structure, as follows:

Value	Meaning
0x00000001	Corresponds to _MONITOR_INFO_1, specified in section 2.2.2.8.1 .
0x00000002	Corresponds to _MONITOR_INFO_2, specified in section 2.2.2.8.2 .

pMonitor: This parameter MAY be null if **cbBuf** equals zero; otherwise, it MUST be a non-null pointer to the [BUFFER](#), as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

BUFFER TYPE: _MONITOR_INFO.

cbBuf: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcReturned: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Additional validation MAY [<233>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<234>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Enumerate all port monitors on the print server.
- Using the enumerated objects, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Return the status of the operation. [<235>](#)

3.1.4.7.2 RpcAddMonitor (Opnum 46)

RpcAddMonitor installs a local port monitor and links the configuration, data, and monitor files.

```
DWORD RpcAddMonitor(
    [in, string, unique] STRING_HANDLE Name,
    [in] MONITOR_CONTAINER* pMonitorContainer
);
```

Name: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pMonitorContainer: This parameter MUST be a non-null pointer to a [MONITOR_CONTAINER](#) structure that MUST contain monitor information. The value of the **Level** member in the **MONITOR_CONTAINER** MUST be two and the **MonitorInfo** member must contain a non-null pointer to a valid structure. For more information about the **MONITOR_CONTAINER** structures, see section [2.2.1.2.6](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Verify that the **pMonitorContainer** parameter is a non-null pointer to a **MONITOR_CONTAINER** with a **Level** member that has a valid value and a **MonitorInfo** member that contains a non-null pointer to a structure. If that verification fails, return **ERROR_INVALID_PARAMETER**, as specified in [MS-ERREF].
- Verify that the port monitor does not already exist in the system, and if that verification fails, return **ERROR_PRINT_MONITOR_ALREADY_INSTALLED**, as specified in [MS-ERREF].
- Additional validation MAY [<236>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<237>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Create the monitor object.
- Return the status of the operation. [<238>](#)

3.1.4.7.3 RpcDeleteMonitor (Opnum 47)

RpcDeleteMonitor removes a port monitor.

```
DWORD RpcDeleteMonitor(
    [in, string, unique] STRING_HANDLE Name,
    [in, string, unique] wchar_t* pEnvironment,
    [in, string] wchar_t* pMonitorName
);
```

Name: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pEnvironment: This parameter MUST adhere to the parameter specification in Environment Name Parameters, section [3.1.4.1.4](#).

pMonitorName: This parameter MUST be a non-null pointer to a string that MUST specify the name of the monitor to remove. For rules governing monitor names, see section [2.2.4.12](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate the following:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in Environment Name Parameters, section [3.1.4.1.4](#).
- Verify that the **pMonitorName** parameter points to a string.

- Verify that there are no ports on the system that are controlled by this monitor at this time.
- Additional validation MAY be performed. [<239>](#)

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<240>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Clear references to the port monitor from any other data structures.
- Delete the port monitor object.
- Return the status of the operation. [<241>](#)

3.1.4.8 Print Processor Management Methods

This section specifies methods for discovering and manipulating print processor objects.

Method	Description
<u>RpcAddPrintProcessor</u>	RpcAddPrintProcessor installs a print processor on the specified server and adds its name to an internal list of supported print processors. Opnum 14
<u>RpcEnumPrintProcessors</u>	RpcEnumPrintProcessors enumerates the print processors installed on a specified server. Opnum 15
<u>RpcGetPrintProcessorDirectory</u>	RpcGetPrintProcessorDirectory retrieves the path for the print processor on the specified server. Opnum 16
<u>RpcDeletePrintProcessor</u>	RpcDeletePrintProcessor removes a print processor. Opnum 48
<u>RpcEnumPrintProcessorDatatypes</u>	RpcEnumPrintProcessorDatatypes enumerates the data types that a specified print processor supports. Opnum 51

3.1.4.8.1 RpcAddPrintProcessor (Opnum 14)

RpcAddPrintProcessor installs a print processor on the specified server and adds its name to an internal list of supported print processors.

```

DWORD RpcAddPrintProcessor(
    [in, string, unique] STRING_HANDLE pName,
    [in, string] wchar_t* pEnvironment,
    [in, string] wchar_t* pPathName,
    [in, string] wchar_t* pPrintProcessorName
);

```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [<3.1.4.1.2>](#).

pEnvironment: This parameter MUST adhere to the parameter specification in Environment Name Parameters, section [3.1.4.1.4](#).

pPathName: This parameter MUST be a non-null pointer to a string that MUST specify the file name of the print processor. For rules governing path names, see section [2.2.4.7](#).

pPrintProcessorName: This parameter MUST be a non-null pointer to a string that MUST specify the name of the print processor. For rules governing print processor names, see section [2.2.4.8](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in Environment Name Parameters, section [3.1.4.1.4](#).
- Verify that the path identified by the string that is referenced by the **pPathName** parameter contains the necessary file for installing the print processor.
- Verify that the print processor to be added does not have the name "winprint", and if that verification fails, return ERROR_PRINT_PROCESSOR_ALREADY_INSTALLED, as specified in [\[MS-ERREF\]](#).
- Additional validation MAY [<242>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<243>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Copy the print processor file as appropriate to its destination and overwrite an existing print processor with the same name, if necessary.
- Create the print processor object.
- If any clients have registered for notification of server object changes, a notification MUST be broadcast to them.
- Return the status of the operation. [<244>](#)

3.1.4.8.2 RpcEnumPrintProcessors (Opnum 15)

RpcEnumPrintProcessors enumerates the print processors installed on a specified server.

```
DWORD RpcEnumPrintProcessors(  
    [in, string, unique] STRING_HANDLE pName,  
    [in, string, unique] wchar_t* pEnvironment,  
    [in] DWORD Level,  
    [in, out, unique, size_is(cbBuf), disable_consistency_check]
```

```

    BYTE* pPrintProcessorInfo,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded,
    [out] DWORD* pcReturned
);

```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pEnvironment: This parameter MUST adhere to the parameter specification in Environment Name Parameters, section [3.1.4.1.4](#).

Level: The value of this parameter MUST be 0x00000001.

pPrintProcessorInfo: This parameter MAY be null if **cbBuf** equals zero; otherwise, it MUST be a non-null pointer to [BUFFER](#) as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#)

BUFFER TYPE: PRINTPROCESSOR_INFO_1.

cbBuf: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcReturned: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in Environment Name Parameters, section [3.1.4.1.4](#).
- Perform the validation steps that are specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<245>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Enumerate all print processors on the print server.
- Using the enumerated objects, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

- Return the status of the operation. [<246>](#)

3.1.4.8.3 RpcGetPrintProcessorDirectory (Opnum 16)

RpcGetPrintProcessorDirectory retrieves the path for the print processor on the specified server.

```
DWORD RpcGetPrintProcessorDirectory(
    [in, string, unique] STRING_HANDLE pName,
    [in, string, unique] wchar_t* pEnvironment,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check]
    BYTE* pPrintProcessorDirectory,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded
);
```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pEnvironment: This parameter MUST adhere to the parameter specification in Environment Name Parameters, section [3.1.4.1.4](#).

Level: The value of this parameter MUST be 0x00000001.

pPrintProcessorDirectory: This parameter MAY be null if cbBuf equals zero; otherwise, it MUST be a non-null pointer to [BUFFER](#) as specified in String Query Parameters, section [3.1.4.1.6](#).

cbBuf: This parameter MUST adhere to the parameter specification in String Query Parameters, section [3.1.4.1.6](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in String Query Parameters, section [3.1.4.1.6](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in Environment Name Parameters, section [3.1.4.1.4](#).
- Perform the validation steps that are specified in String Query Parameters, section [3.1.4.1.6](#).
- Additional validation MAY [<247>](#) be performed.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<248>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Using the path of the print processor directory on the print server, perform the processing and response steps specified in String Query Parameters, section [3.1.4.1.6](#).
- Return the status of the operation. [<249>](#)

3.1.4.8.4 RpcDeletePrintProcessor (Opnum 48)

RpcDeletePrintProcessor removes a print processor.

```
DWORD RpcDeletePrintProcessor(
    [in, string, unique] STRING_HANDLE Name,
    [in, string, unique] wchar_t* pEnvironment,
    [in, string] wchar_t* pPrintProcessorName
);
```

Name: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pEnvironment: This parameter MUST adhere to the parameter specification in Environment Name Parameters, section [3.1.4.1.4](#).

pPrintProcessorName: This parameter MUST be a non-null pointer to a string that MUST specify the name of the print processor that MUST be removed. For rules governing print processor names, see PrintProcessorName, section [2.2.4.8](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate the following:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Perform the validation steps that are specified in Environment Name Parameters, section [3.1.4.1.4](#).
- The string that is referenced by the **pPrintProcessorName** parameter identifies a print processor installed on the server.
- Verify that there are no printers on the system that use the print processor at this time.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<250>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Clear all references to the specified print processor from any other data structures.
- Delete the print processor object.
- If any clients have registered for notification of server object changes, a notification MUST be broadcast to them.

- Return the status of the operation. [<251>](#)

3.1.4.8.5 RpcEnumPrintProcessorDatatypes (Opnum 51)

RpcEnumPrintProcessorDatatypes enumerates the data types that a specified print processor supports.

```
DWORD RpcEnumPrintProcessorDatatypes(
    [in, string, unique] STRING_HANDLE pName,
    [in, string, unique] wchar_t* pPrintProcessorName,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check]
    BYTE* pDatatypes,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded,
    [out] DWORD* pcReturned
);
```

pName: This parameter MUST adhere to the parameter specification in Print Server Name Parameters, section [3.1.4.1.2](#).

pPrintProcessorName: This parameter MUST be a non-null pointer to a string that MUST specify the name of the print processor whose data types MUST be enumerated. For rules governing print processor names, see section [2.2.4.8](#).

Level: The value of this parameter MUST be 0x00000001.

pDatatypes: This parameter MAY be null if **cbBuf** equals zero; otherwise, it MUST be a non-null pointer to [BUFFER](#) as specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).

BUFFER TYPE: `_DATATYPES_INFO_1`

cbBuf: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcbNeeded: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

pcReturned: This parameter MUST adhere to the parameter specification in INFO Structures Query Parameters, section [3.1.4.1.5](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure.

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in Print Server Name Parameters, section [3.1.4.1.2](#).
- Verify that the print processor that is identified by the string that is referenced by the **pPrintProcessorName** parameter is in the list of print processors.

- Perform the validation steps that are specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Additional validation MAY [<252>](#) be performed.
- If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<253>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Enumerate all data types that are supported by the specified print processor.
- Using the enumerated objects, perform the processing and response steps specified in INFO Structures Query Parameters, section [3.1.4.1.5](#).
- Return the status of the operation. [<254>](#)

3.1.4.9 Document Printing Methods

This section specifies methods for printing documents, pages and data.

Method	Description
RpcStartDocPrinter	RpcStartDocPrinter notifies the print spooler that a document is being spooled for printing. Opnum 17
RpcStartPagePrinter	RpcStartPagePrinter notifies the spooler that a page is about to be printed on the specified printer. Opnum 18
RpcWritePrinter	RpcWritePrinter sends data to the print spooler. Opnum 19
RpcEndPagePrinter	RpcEndPagePrinter notifies the print spooler that the application is at the end of a page in a print job. Opnum 20
RpcAbortPrinter	RpcAbortPrinter aborts the currently spooling print document. Opnum 21
RpcReadPrinter	RpcReadPrinter retrieves data from the specified printer. Opnum 22
RpcEndDocPrinter	RpcEndDocPrinter notifies the print spooler that the application is at the end of the current print job. Opnum 23
RpcFlushPrinter	RpcFlushPrinter is used by the printer driver to send a buffer of bytes to the port to cleanly abort a print job. It also allows delaying the I/O line to the printer. Opnum 96

3.1.4.9.1 RpcStartDocPrinter (Opnum 17)

RpcStartDocPrinter notifies the print server that a document is being spooled for printing.

```

DWORD RpcStartDocPrinter(
    [in] PRINTER_HANDLE hPrinter,
    [in] DOC_INFO_CONTAINER* pDocInfoContainer,
    [out] DWORD* pJobId
);

```

hPrinter: This parameter MUST specify a handle to a printer object or port object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods. The printer handle MUST NOT be in use for printing another document at the time of this call.

pDocInfoContainer: This parameter MUST adhere to the validation specified for [DOC_INFO_CONTAINER Parameters](#), section [3.1.4.1.9.2](#).

pJobId: This parameter MUST be a non-null pointer to a variable that MUST receive a nonzero print job identifier. The job MUST be created with an identifier that is unique for this printer.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate the following:

- Perform the validation steps that are specified in [PRINTER_HANDLE Parameters](#), section [3.1.4.1.10](#).
- Perform the validation steps that are specified in [DOC_INFO_CONTAINER Parameters](#), section [3.1.4.1.9.2](#).
- The server MUST verify that **RpcStartDocPrinter** does not get called twice without intervening [RpcEndDocPrinter](#); and if that verification fails, return [ERROR_INVALID_HANDLE](#), as specified in [\[MS-ERREF\]](#).

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<255>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Create the job object.
- Write the ID of the created job in the variable that is pointed to by the **pJobId** parameter.
- If any clients that have registered for notification of the job object creation, a notification MUST be broadcast to them.
- Return the status of the operation. [<256>](#)

3.1.4.9.2 RpcStartPagePrinter (Opnum 18)

RpcStartPagePrinter notifies the spooler that a page is about to be printed on the specified printer.

```

DWORD RpcStartPagePrinter(

```

```
[in] PRINTER_HANDLE hPrinter
);
```

hPrinter: This parameter MUST specify a handle to a printer object or port object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate the following:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that a job has been associated with **hPrinter** using [RpcStartDocPrinter](#), and if that verification fails, return ERROR_SPL_NO_STARTDOC, as specified in [\[MS-ERREF\]](#).
- Verify that printing of the job has not been canceled, and if that verification fails, return ERROR_PRINT_CANCELLED, as specified in [\[MS-ERREF\]](#).
- The server MUST NOT enforce that **RpcStartPagePrinter** and [RpcEndPagePrinter](#) are called in balanced pairs.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<257>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Update print job statistics to reflect incremented page count.
- Return the status of the operation. [<258>](#)

3.1.4.9.3 RpcWritePrinter (Opnum 19)

RpcWritePrinter sends data to the print server.

```
DWORD RpcWritePrinter(
    [in] PRINTER_HANDLE hPrinter,
    [in, size_is(cbBuf)] BYTE* pBuf,
    [in] DWORD cbBuf,
    [out] DWORD* pcWritten
);
```

hPrinter: This parameter MUST specify a handle to a printer object or port object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pBuf: This parameter MAY be null if the value of the **cbBuf** parameter is zero or it MUST be a non-null pointer to a buffer that contains the data to be written.

cbBuf: The value of this parameter MUST be the number of bytes of data to be written.

pcWritten: This parameter MUST be a non-null pointer to a value that MUST receive the number of bytes of data that were written.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that a job has been associated with **hPrinter** using [RpcStartDocPrinter](#), and if that verification fails, return ERROR_SPL_NO_STARTDOC, as specified in [\[MS-ERREF\]](#).
- Verify that printing of the job has not been canceled, and if that verification fails, return ERROR_PRINT_CANCELLED, as specified in [\[MS-ERREF\]](#).
- If **cbBuf** is not zero, verify that **pBuf** is not null.
- The server MUST NOT enforce that [RpcStartPagePrinter](#) has been called before calling **RpcWritePrinter**.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client.[<259>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- If the **hPrinter** parameter is a printer object handle, copy **cbBuf** bytes of data pointed to by **pBuf** to the job; depending on server policy and settings, the data is added to temporary storage of the job (for example, a spool file), or sent directly to the port.
- If the **hPrinter** parameter is a port object handle, copy **cbBuf** bytes of data pointed to by **pBuf** directly to the port.
- Write the number of bytes that were written to the variable that is pointed to by the **pcWritten** parameter.
- Return the status of the operation.
- If the operation is successful, the server MUST modify the job object to indicate the number of bytes written so far to that job.

3.1.4.9.4 RpcEndPagePrinter (Opnum 20)

RpcEndPagePrinter notifies the print server that the application is at the end of a page in a print job.

```
DWORD RpcEndPagePrinter(  
    [in] PRINTER_HANDLE hPrinter  
);
```

hPrinter: This parameter MUST specify a handle to a printer object or port object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that printing of the job has not been canceled, and if that verification fails, return ERROR_PRINT_CANCELLED, as specified in [\[MS-ERREF\]](#).
- The server MUST NOT enforce that [RpcStartPagePrinter](#) and **RpcEndPagePrinter** are called in balanced pairs.
- If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<260>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- This method MAY trigger scheduling the job to the device, depending on server settings and policy, such as "Print while spooling."
- Update the count of the pages processed and printed so far in the job object.
- If any clients have registered for notification of job object changes, a notification MUST be broadcast to them.
- Return the status of the operation. [<261>](#)

3.1.4.9.5 RpcAbortPrinter (Opnum 21)

RpcAbortPrinter aborts the currently spooling print document.

```
DWORD RpcAbortPrinter(  
    [in] PRINTER_HANDLE hPrinter  
);
```

hPrinter: This parameter MUST specify a handle to a printer object or port object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in **PRINTER_HANDLE** Parameters, section [3.1.4.1.10](#).
- Verify that a job has been associated with **hPrinter** by using [RpcStartDocPrinter](#), and if that verification fails, return **ERROR_SPL_NO_STARTDOC**, as specified in [MS-ERREF].
- If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<262>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- The current job is aborted. If it is in spool stage, spooling MUST stop. If it is in printing stage, printing MUST stop.
- Modify the job object to indicate that the job has been aborted.
- Delete the spool file, if one exists.
- Delete the job object.
- Modify the list of jobs to exclude this deleted job.
- Return the status of the operation. [<263>](#)

3.1.4.9.6 **RpcReadPrinter (Opnum 22)**

RpcReadPrinter retrieves data from the specified job or port.

```
DWORD RpcReadPrinter(
    [in] PRINTER_HANDLE hPrinter,
    [out, size_is(cbBuf)] BYTE* pBuf,
    [in] DWORD cbBuf,
    [out] DWORD* pcNoBytesRead
);
```

hPrinter: This parameter MUST specify a handle to a job object or port object that MUST have been opened using the [RpcOpenPrinter](#) or [RpcOpenPrinterEx](#) methods.

pBuf: This parameter MAY be null if the value of the **cbBuf** parameter is zero or it MUST be a non-null pointer to a buffer that MUST receive the printer data. If the **hPrinter** parameter is the handle to a port object, this method returns the data that is returned by the port monitor.

cbBuf: The value of this parameter MUST be the size, in bytes, of data to be read into the buffer that is pointed to by the **pBuf** parameter.

pcNoBytesRead: This parameter MUST be a non-null pointer to a variable that MUST receive the number of bytes of data copied into the array to which **pBuf** points.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in [PRINTER_HANDLE Parameters](#), section [3.1.4.1.10](#).
- Verify that printing of the job has not been canceled and if that verification fails, return `ERROR_PRINT_CANCELLED`, as specified in [MS-ERREF].
- If the value of the **cbBuf** parameter is not zero, verify that the **pBuf** parameter is not null.
- If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<264>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- If the **hPrinter** parameter is a job object handle, copy data from the temporary storage of the job object to the buffer pointed to by **pBuf**, up to the number of bytes indicated in **cbBuf**, or to the end of the temporary storage's data, whichever comes first.
- If the **hPrinter** parameter is a port object handle, read directly from the port and copy the read data to the buffer pointed to by **pBuf**, up to the number of bytes indicated in **cbBuf**, or no more data can be read, whichever comes first.
- Write the number of bytes that were copied to the variable that is pointed to by **pcNoBytesRead**.
- If reading from a job object, update the read pointer, so a subsequent read will continue at the correct location.
- Return the status of the operation. [<265>](#)

3.1.4.9.7 RpcEndDocPrinter (Opnum 23)

RpcEndDocPrinter notifies the print server that the application is at the end of the current print job.

```
DWORD RpcEndDocPrinter(
    [in] PRINTER_HANDLE hPrinter
);
```

hPrinter: This parameter MUST specify a handle to a printer object or port object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

Return value/code	Description
0x00000000 <code>ERROR_SUCCESS</code>	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in [PRINTER_HANDLE Parameters](#), section [3.1.4.1.10](#).
- Verify that a job has been associated with **hPrinter** by using [RpcStartDocPrinter](#), and if that verification fails, return `ERROR_SPL_NO_STARTDOC`, as specified in [MS-ERREF].

- If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<266>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- If any clients have registered for notification of job object changes, a notification MUST be broadcast to them.
- Modify the state of the job object to indicate the job has completed spooling if **hPrinter** is a printer object handle; or has completed printing if **hPrinter** is a port object handle.
- If the **hPrinter** parameter is a printer object handle, schedule job for printing (subject to configuration of the server or implementation-specific policies) and modify the state of the job object to indicate the job is printing.
- Return the status of the operation. [<267>](#)

3.1.4.9.8 RpcFlushPrinter (Opnum 96)

RpcFlushPrinter is used by printer drivers to send a buffer of bytes to the port to cleanly abort a print job. It also allows delaying the I/O line to the printer.

```
DWORD RpcFlushPrinter(
    [in] PRINTER_HANDLE hPrinter,
    [in, size_is(cbBuf)] BYTE* pBuf,
    [in] DWORD cbBuf,
    [out] DWORD* pcWritten,
    [in] DWORD cSleep
);
```

hPrinter: This parameter MUST specify a handle to a port object that MUST have been opened by using the [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

pBuf: This parameter MAY be null if the value of the **cbBuf** parameter is zero or else it MUST be a non-null pointer to the array of bytes containing the data to be written to the printer.

cbBuf: The value of this parameter MUST specify the size, in bytes, of the array pointed to by the **pBuf** parameter.

pcWritten: This parameter MUST be a non-null pointer to a variable that MUST receive the number of bytes of data that were written to the printer.

cSleep: The value of this parameter MUST specify the time, in milliseconds, that the I/O line to the printer port MUST be delayed. A value of zero MUST result in no delay.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure.

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).

- Verify that a previous [RpcWritePrinter](#) has failed due to job cancellation, and if that verification fails, return **ERROR_INVALID_HANDLE** as specified in [\[MS-ERREF\]](#).
- If the value of the **cbBuf** parameter is not zero, verify that the **pBuf** parameter is not null.
- If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [.<268>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Send the contents of the buffer that is pointed to by the **pBuf** parameter to the port.
- If the value of the **cSleep** parameter is not zero, the server MUST halt operations to the port for the number of milliseconds specified by the value of the **cSleep** parameter.
- Write the number of bytes that were written to the port, to the variable that is pointed to by the **pcWritten** parameter.
- Return the status of the operation. [.<269>](#)

3.1.4.10 Notification Methods

This section specifies methods for obtaining notifications of printing events.

Method	Description
RpcWaitForPrinterChange	RpcWaitForPrinterChange retrieves information about the most recent change notification associated with a printer or print server. Opnum 28
RpcFindClosePrinterChangeNotification	RpcFindClosePrinterChangeNotification closes a change notification object created by calling either the RpcRemoteFindFirstPrinterChangeNotification or RpcRemoteFindFirstPrinterChangeNotificationEx function. The printer or print server associated with the change notification object will no longer be monitored by that object. Opnum 56
RpcRemoteFindFirstPrinterChangeNotification	RpcRemoteFindFirstPrinterChangeNotification creates a remote change notification object that monitors changes to printer objects, and sends change notifications to the client using the method RpcRouterReplyPrinter (see section 3.2.4.1.2). Opnum 62
RpcRemoteFindFirstPrinterChangeNotificationEx	RpcRemoteFindFirstPrinterChangeNotificationEx creates a remote change notification object that monitors changes to printer objects, and sends change notifications to the client using the method RpcRouterReplyPrinterEx (see section 3.2.4.1.4). Opnum 65
RpcRouterRefreshPrinterChangeNotification	RpcRouterRefreshPrinterChangeNotification

Method	Description
	returns change notification information. Opnum 67

3.1.4.10.1 RpcWaitForPrinterChange (Opnum 28)

RpcWaitForPrinterChange retrieves information about the most recent change notification that is associated with a printer or print server.

```

DWORD RpcWaitForPrinterChange(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD Flags,
    [out] DWORD* pFlags
);

```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

Flags: The value of this parameter MUST specify the change notifications to wait for. The value of this parameter MUST be the result of a bitwise OR of one or more printer change values. Printer change values are defined in section [2.2.3.3](#). For rules governing them, see section [2.2.4.10](#).

pFlags: This parameter MUST be a non-null pointer to a variable that MUST receive the bitwise OR combination of one or more printer change values. Printer change values are defined in section [2.2.3.3](#). For rules governing them, see section [2.2.4.10](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).

If parameter validation fails, the server MUST [<270>](#) fail the operation immediately, returning a nonzero error response to the client.

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Wait for an implementation-specific period of time, [<271>](#) or until one of the changes specified by the value of the **Flags** parameter occurs.
- If one or more of the specified changes occurred within the time-out period, write a bitwise OR combination of the changes to the variable that is pointed to by **pFlags** and return zero.
- If the time-out period has expired without any of the specified changes, return PRINTER_CHANGE_TIMEOUT, as specified in [\[MS-ERREF\]](#).

Note: Because this method waits for an implementation-specific, potentially long, period of time, it can cause the client system to stop responding. Therefore, this method is deprecated and SHOULD NOT be used. The implementer of a protocol client SHOULD consider using [RpcRemoteFindFirstPrinterChangeNotificationEx](#) instead.

3.1.4.10.2 RpcFindClosePrinterChangeNotification (Opnum 56)

The **RpcFindClosePrinterChangeNotification** method closes a change notification object created by calling the [RpcRemoteFindFirstPrinterChangeNotificationEx](#) function. The printer or print server associated with the change notification object will no longer be monitored by that object.

```
DWORD RpcFindClosePrinterChangeNotification(
    [in] PRINTER_HANDLE hPrinter
);
```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform validation steps as specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that there is a change notification object associated with the printer object handle.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<272>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Clear all internal change notification objects associated with the **hPrinter**.
- Return the status of the operation. [<273>](#)

3.1.4.10.3 RpcRemoteFindFirstPrinterChangeNotification (Opnum 62)

RpcRemoteFindFirstPrinterChangeNotification creates a remote change notification object that monitors changes to printer objects, and sends change notifications to the client using the method [RpcRouterReplyPrinter](#) (see section [3.2.4.1.2](#)). [<274>](#)

```
DWORD RpcRemoteFindFirstPrinterChangeNotification(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD fdwFlags,
    [in] DWORD fdwOptions,
    [in, string, unique] wchar_t* pszLocalMachine,
    [in] DWORD dwPrinterLocal,
    [in, range(0, 512)] DWORD* cbBuffer,
    [in, out, unique, size_is(cbBuffer), disable_consistency_check]
    BYTE* pBuffer
```

);

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

fdwFlags: The value of this parameter MUST specify the conditions that MUST be met for a change notification object to enter a signaled state. A change notification MUST occur when one or more of the specified conditions are met. This parameter MUST specify a bitwise OR of zero or more printer change values. Printer change values are defined in section [2.2.3.3](#). The rules governing them are specified in section [2.2.4.10](#).

fdwOptions: This parameter is reserved. The value of this parameter MUST be set to 0x00000000 and this parameter MUST be ignored upon receipt.

pszLocalMachine: This parameter MUST be a non-null pointer to a string that MUST represent the name of the client computer. For rules governing server names, see section [2.2.4.1](#).

dwPrinterLocal: The value of this parameter MUST specify an implementation-specific unique value. This value MUST be sufficient for the client to identify a call to [RpcReplyOpenPrinter](#) that comes back from a server as being a call that is associated with the **hPrinter** parameter in this call. [<275>](#)

cbBuffer: This parameter MUST specify the size in bytes of the buffer pointed to by **pBuffer**.

pBuffer: This parameter MUST be ignored.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps as specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that a notification object is not already associated with the current handle.

If parameter validation fails, the server MUST fail the operation immediately returning a nonzero error response to the client. [<276>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Create and initialize a notification object that captures the notification settings requested by the user.
- Create and initialize a notification channel back to the client over which the server MUST communicate the change notifications. This MUST be done by calling [RpcReplyOpenPrinter](#) on the client specified by the name pointed to by **pszLocalMachine**.
- Associate the notification object with the context for **hPrinter**.
- After the preceding steps have been performed, the server SHOULD call the client's **RpcRouterReplyPrinter** when monitored objects change.

- Return the status of the operation. [<277>](#)

3.1.4.10.4 RpcRemoteFindFirstPrinterChangeNotificationEx (Opnum 65)

RpcRemoteFindFirstPrinterChangeNotificationEx creates a remote change notification object that monitors changes to printer objects, and sends change notifications to the client using the method [RpcRouterReplyPrinterEx](#) (see section [3.2.4.1.4](#)).

```
DWORD RpcRemoteFindFirstPrinterChangeNotificationEx(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD fdwFlags,
    [in] DWORD fdwOptions,
    [in, string, unique] wchar_t* pszLocalMachine,
    [in] DWORD dwPrinterLocal,
    [in, unique] RPC_V2_NOTIFY_OPTIONS* pOptions
);
```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods.

fdwFlags: The value of this parameter MUST specify the conditions that MUST be met for a change notification object to enter a signaled state. A change notification MUST occur when one or more of the specified conditions are met. This parameter MUST specify a bitwise OR of zero or more printer change values. Printer change values are defined in section [2.2.3.3](#). The rules governing them are specified in section [2.2.4.10](#).

fdwOptions: This parameter is reserved. The value of this parameter MUST be set to 0x00000000 and this parameter MUST be ignored upon receipt.

pszLocalMachine: This parameter MUST be a non-null pointer to a string that MUST represent the name of the client computer. For rules governing server names, see section [2.2.4.1](#).

dwPrinterLocal: The value of this parameter MUST specify an implementation-specific unique value. This value MUST be sufficient for the client to identify a call to [RpcReplyOpenPrinter](#) that comes back from a server as being a call that is associated with the **hPrinter** parameter in this call. [<278>](#)

pOptions: This parameter MAY be null if the value of the **fdwFlags** parameter is nonzero; otherwise, it MUST be a non-null pointer to an [RPC_V2_NOTIFY_OPTIONS](#) structure that MUST specify the printer or the job members that the client wants to listen for notifications on. For information on members, see printer notification values in section [2.2.3.5](#) and job notification values in section [2.2.3.6](#).

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps as specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that a notification object is not already associated with the current handle.

- Verify that either **pOptions** is not null or that the value **fdwFlags** is valid and not zero.

If parameter validation fails, the server MUST fail the operation immediately returning a nonzero error response to the client. [<279>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Create and initialize a notification object that captures the notification settings requested by the user.
- Create and initialize a notification channel back to the client over which the server MUST communicate the change notifications. This MUST be done by calling `RpcReplyOpenPrinter` on the client specified by the name pointed to by **pszLocalMachine**.
- Associate the notification object with the context for **hPrinter**.
- After the preceding steps have been performed, the server SHOULD call the client's `RpcRouterReplyPrinterEx` when monitored objects change.
- Return the status of the operation. [<280>](#)

3.1.4.10.5 RpcRouterRefreshPrinterChangeNotification (Opnum 67)

RpcRouterRefreshPrinterChangeNotification returns change notification information.

```
DWORD RpcRouterRefreshPrinterChangeNotification(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD dwColor,
    [in, unique] RPC_V2_NOTIFY_OPTIONS* pOptions,
    [out] RPC_V2_NOTIFY_INFO** ppInfo
);
```

hPrinter: This parameter MUST specify a handle to a printer object or server object that MUST have been opened by using the [RpcAddPrinter](#), [RpcAddPrinterEx](#), [RpcOpenPrinter](#), or [RpcOpenPrinterEx](#) methods, and on which the client previously successfully called [RpcRemoteFindFirstPrinterChangeNotification](#) or [RpcRemoteFindFirstPrinterChangeNotificationEx](#) and has not since closed the notification state by calling [RpcFindClosePrinterChangeNotification](#).

dwColor: The value of this parameter MUST be an implementation-specific client context value that is used by the clients to get an indication of notifications order. [<281>](#)

pOptions: This parameter MUST be a non-null pointer to a [RPC_V2_NOTIFY_OPTIONS](#) structure. For a list of fields that can be monitored, see printer notification values in section [2.2.3.5](#) and job notification values in section [2.2.3.6](#).

ppInfo: This parameter MUST be a non-null pointer to a variable that MUST receive a pointer to an [RPC_V2_NOTIFY_INFO](#) structure that MUST contain notification information.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the server MUST validate parameters as follows:

- Perform the validation steps that are specified in PRINTER_HANDLE Parameters, section [3.1.4.1.10](#).
- Verify that a notification back channel to the client has been established and is still open.

If parameter validation fails, the server MUST fail the operation immediately and return a nonzero error response to the client. [<282>](#)

Otherwise, the server MUST process the message and compose a response to the client as follows:

- Collect all the notification data requested for the printer objects.
- Allocate a buffer and write the collected notification data in the buffer.
- Associate the buffer with **ppInfo** output parameter.
- Return the status of the operation. [<283>](#)

This method MUST be called when the client receives a PRINTER_NOTIFY_INFO structure with the PRINTER_NOTIFY_INFO_DISCARDED bit set in the Flags member, which indicates that an overflow or other error has occurred, and that notifications might have been lost, setting the notification object to the discarded state. In that case, the server MUST NOT send any additional notifications until the client makes this method call. If the operation is successful, the server MUST modify the notification object to clear the discarded state.

3.1.4.11 Monitor Module Methods

A **monitor module** is a server-side executable object that provides a communication path between a print server and the drivers that access hardware on a machine. A port monitor module manages access to I/O port hardware.

Port monitor modules are implementation specific for a given port type.

Port monitor modules MUST support the following methods:

- AddPort
- ConfigurePort
- DeletePort

Port monitor modules SHOULD support an additional method, **XcvData**, which SHOULD support the following actions (see section [3.1.4.6.5](#)):

- AddPort
- DeletePort
- MonitorUI

Actions MUST be specified by the client in a string pointed to by the **pszDataName** parameter of **RpcXcvData**.

Additional actions MAY [<284>](#) be supported in a given implementation. For historical reasons, the names of some of the actions supported by **XcvData** are identical to some of the other port monitor

module methods. The server method **RpcXcvData** routes calls to a port monitor's **XcvData** method, and the parameter lists of **RpcXcvData** and the port monitor's **XcvData** are identical.

3.1.4.11.1 Localmon

Information about the implementation of **XcvData** methods in the **localmon** monitor module can be found in [Appendix B: Windows Behavior.<285>](#)

3.1.4.11.2 Lprmon

Information about the implementation of **XcvData** methods in the **lprmon** can be found in [Appendix B: Windows Behavior.<286>](#)

3.1.4.11.3 Tcpmon

Information about the implementation of the **tcpmon** monitor module can be found in the following subsections and in [Appendix B: Windows Behavior.<287>](#)

3.1.4.11.3.1 Structures

3.1.4.11.3.1.1 PORT_DATA_1

Information about **PORT_DATA_1** can be found in [Appendix B: Windows Behavior.<288>](#)

3.1.4.11.3.1.2 DELETE_PORT_DATA_1

Information about **DELETE_PORT_DATA_1** can be found in [Appendix B: Windows Behavior.<289>](#)

3.1.4.11.3.1.3 CONFIG_INFO_DATA_1

Information about **CONFIG_INFO_DATA_1** can be found in [Appendix B: Windows Behavior.<290>](#)

3.1.4.11.3.1.4 PORT_DATA_LIST_1

Information about **PORT_DATA_LIST_1** can be found in [Appendix B: Windows Behavior.<291>](#)

3.1.4.11.3.1.5 PORT_DATA_2

Information about **PORT_DATA_2** can be found in [Appendix B: Windows Behavior.<292>](#)

3.1.4.11.3.2 Command Value Table

Information about command values used by the tcpmon monitor module can be found in [Appendix B: Windows Behavior.<293>](#)

3.1.4.11.4 Wsdmon

Information about [WSD_DRIVER_DATA](#) can be found in [Appendix B: Windows Behavior.<294>](#)

3.1.4.11.4.1 Structures

3.1.4.11.4.1.1 WSD_DRIVER_DATA

Information about **WSD_DRIVER_DATA** can be found in [Appendix B: Windows Behavior.<295>](#)

3.1.4.11.4.1.2 WSD_BACKUP_PORT_DATA

Information about WSD_BACKUP_PORT_DATA can be found in [Appendix B: Windows Behavior.<296>](#)

3.1.4.11.4.2 Command Value Table

Information about [WSD_DRIVER_DATA](#) can be found in [Appendix B: Windows Behavior.<297>](#)

3.1.5 Timer Events

No protocol timer events are required on the client beyond the timers required in the underlying RPC protocol.

3.1.6 Other Local Events

No additional local events are used on the client beyond the events maintained in the underlying RPC protocol.

3.2 Client Details

3.2.1 Abstract Data Model

No abstract data model is required.

3.2.2 Timers

No protocol timers are required beyond those used internally by RPC to implement resiliency to network outages, as specified in [\[MS-RPCE\]](#) section 3.2.3.2.

3.2.3 Initialization

The client MUST perform initialization according to the following rules when calling an RPC method:

- Either create an RPC binding handle to the server or use an RPC context handle. Details concerning binding handles are as specified in [\[C706\]](#) section 2.3.
- Use context handles across multiple calls to the server for methods taking a [PRINTER_HANDLE](#).
- Use handles bound to a single call to the server for name-based methods taking a [STRING_HANDLE](#). A **STRING_HANDLE_BIND** method MUST be implemented by the client.
- Reuse a context handle in multiple invocations when creating a print job, such as in a call to [RpcOpenPrinter](#) followed by multiple calls to [RpcStartPagePrinter](#) and [RpcWritePrinter](#). For an example of this sequence of calls, see section [3.2.4.2.1](#).
- A context handle SHOULD be reused in multiple invocations when getting or setting information on a printer, such as in a call to **RpcOpenPrinter** followed by multiple calls to [RpcGetPrinter](#), [RpcGetPrinterData](#), [RpcSetPrinter](#), or other methods taking a **PRINTER_HANDLE** or [GDI_HANDLE](#).
- When creating the RPC binding handle on the named pipe `\pipe\spoolss`, the client MUST specify an **ImpersonationLevel** of 2 (**Impersonation**), as specified in [\[MS-SMB2\]](#) section 2.2.13.

3.2.4 Message Processing Events and Sequencing Rules

This protocol MUST indicate to the RPC runtime that:

- It is to perform a strict NDR data consistency check at target level 6.0; and
- It is to reject a NULL unique or full pointer with a nonzero conformant value;

as specified in [\[MS-RPCE\]](#) section 3.

The client SHOULD ignore errors returned from the RPC server and SHOULD notify the application invoker of the error received in the higher layer. Unless otherwise specified, no special message processing is required on the client beyond that required in the underlying RPC protocol. [<298>](#)

3.2.4.1 Client-Side Notification-Processing Methods

This section contains the processing rules for the notification-processing methods that the client system MUST implement to handle notifications from the print server.

- Client-Side Notification-Processing:
 - [RpcReplyOpenPrinter](#)
 - [RpcRouterReplyPrinter](#)
 - [RpcReplyClosePrinter](#)
 - [RpcRouterReplyPrinterEx](#)

All these methods are request/response RPC methods. They MUST return zero to indicate successful completion and nonzero values to indicate failure, except where specifically described.

3.2.4.1.1 RpcReplyOpenPrinter (Opnum 58)

RpcReplyOpenPrinter establishes a context handle from the server to the client. The server uses the RPC context handle returned by this method to send notification data to the client machine.

```
DWORD RpcReplyOpenPrinter(  
    [in, string] STRING_HANDLE pMachine,  
    [out] PRINTER_HANDLE* phPrinterNotify,  
    [in] DWORD dwPrinterRemote,  
    [in] DWORD dwType,  
    [in, range(0, 512)] DWORD cbBuffer,  
    [in, unique, size_is(cbBuffer), disable_consistency_check]  
    BYTE* pBuffer  
);
```

pMachine: This parameter is synonymous with **pName**, as specified in [Print Server Name Parameters \(section 3.1.4.1.2\)](#), which specifies the client computer name.

phPrinterNotify: This parameter MUST be a non-null pointer to a remote printer RPC context handle that MUST receive the handle used by the server to send notification to the client computer. RPC context handles are specified in [\[C706\]](#) section [2.3](#).

dwPrinterRemote: The value of this parameter MUST be the value that is supplied to the server by the **dwPrinterLocal** parameter of the corresponding

[RpcRemoteFindFirstPrinterChangeNotification](#) or [RpcRemoteFindFirstPrinterChangeNotificationEx](#) call. This value MUST NOT be zero.

dwType: The value of this parameter MUST be 0x00000001.

cbBuffer: The value of this parameter SHOULD be zero and this parameter MUST be ignored upon receipt.

pBuffer: This parameter SHOULD be set to null and it MUST be ignored upon receipt.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion. Note If the method fails, a return value of nonzero indicates failure, as a Windows error code.

Upon receiving this message, the client MUST validate parameters as follows:

- Verify that the **pMachine** parameter corresponds to the current machine.

If parameter validation fails, the client MUST fail the operation immediately and return a nonzero error response to the server. [<299>](#)

Otherwise, the client MUST process the message as follows:

- Locate the notification state that is identified by the **dwPrinterRemote** parameter.
- Create a back channel RPC context handle and associate it with this notification state.
- Store the back channel RPC context handle in the handle pointed to by **phPrinterNotify**.
- Return the status of the operation. [<300>](#)

3.2.4.1.2 RpcRouterReplyPrinter (Opnum 59)

RpcRouterReplyPrinter handles the notification coming from a server.

```
DWORD RpcRouterReplyPrinter(  
    [in] PRINTER_HANDLE hNotify,  
    [in] DWORD fdwFlags,  
    [in, range(0, 512)] DWORD cbBuffer,  
    [in, unique, size_is(cbBuffer), disable_consistency_check]  
    BYTE* pBuffer  
);
```

hNotify: This parameter MUST be a notification handle and it MUST have been opened by the server by calling [RpcReplyOpenPrinter](#).

fdwFlags: The value of this parameter MUST specify the flags that indicate what values have changed. The value of this parameter MUST be the result of a bitwise OR of one or more printer change values. Printer change values are specified in section [2.2.3.3](#). The rules governing them MUST be as specified in section [2.2.4.10](#).

cbBuffer: The value of this parameter SHOULD be zero and it MUST be ignored upon receipt.

pBuffer: This parameter SHOULD be null and it MUST be ignored upon receipt.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the client MUST validate parameters as follows:

- Verify that the **hNotify** parameter is an RPC context handle to a notification object opened using `RpcReplyOpenPrinter`, and if that verification fails, return `ERROR_INVALID_HANDLE`, as specified in [\[MS-ERREF\]](#).

If parameter validation fails, the client MUST fail the operation immediately and return a nonzero error response to the server. [<301>](#)

Otherwise, the client MUST process the message as follows:

- Capture the **fdwFlags** in the notification state it maintains.

If the operation is successful, the client MUST send the received data to the caller that registered for the notifications, using [RpcRemoteFindFirstPrinterChangeNotification](#) or [RpcRemoteFindFirstPrinterChangeNotificationEx](#).

3.2.4.1.3 RpcReplyClosePrinter (Opnum 60)

RpcReplyClosePrinter closes the notification channel opened by the [RpcReplyOpenPrinter](#) method (see section [3.2.4.1.1](#)) between the server and client.

```
DWORD RpcReplyClosePrinter(  
    [in, out] PRINTER_HANDLE* phNotify  
);
```

phNotify: This parameter MUST be a non-null pointer to the notification context handle that this method MUST close. The handle MUST have been opened by calling **RpcReplyOpenPrinter**.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the client MUST validate parameters as follows:

- Verify that the **hNotify** parameter is an RPC context handle to a notification object that was opened using **RpcReplyOpenPrinter**, and if that verification fails, return `ERROR_INVALID_HANDLE`, as specified in [\[MS-ERREF\]](#).

If parameter validation fails, the client MUST fail the operation immediately and return a nonzero error response to the server. [<302>](#)

Otherwise, the client MUST process the message as follows:

- Free the context handle associated with the notification state.

- Return a response to the client containing the output parameters and the status of the operation.

If the operation is successful, the client **MUST** modify the notification state by removing the back channel context handle associated with it.

3.2.4.1.4 RpcRouterReplyPrinterEx (Opnum 66)

RpcRouterReplyPrinterEx handles the notification coming from a print server.

```
DWORD RpcRouterReplyPrinterEx(
    [in] PRINTER_HANDLE hNotify,
    [in] DWORD dwColor,
    [in] DWORD fdwFlags,
    [out] DWORD* pdwResult,
    [in] DWORD dwReplyType,
    [in, switch_is(dwReplyType)] RPC_V2_UREPLY_PRINTER Reply
);
```

hNotify: This parameter **MUST** be a handle to a notification context handle. The handle **MUST** have been opened by calling [RpcReplyOpenPrinter](#).

dwColor: The value of this parameter **MUST** be the same value that was previously passed by the client in the most recent call to [RpcRouterRefreshPrinterChangeNotification](#). For details on this value, see section [3.1.4.10.5](#).

fdwFlags: The value of this parameter **MUST** specify the flags that indicate the changed printer values. The value of this parameter **MUST** be the result of a bitwise OR of one or more printer change values. Printer change values are specified in section [2.2.3.3](#). The rules governing them are specified in section [2.2.4.10](#).

pdwResult: This parameter **MUST** be a non-null pointer to a variable that **MUST** receive the result of a bitwise OR of the flags that indicate how the client processed the notification. Change notification flag values are specified in section [2.2.3.7](#).

dwReplyType: The value of this parameter **MUST** be zero.

Reply: This parameter **MUST** be a non-null pointer to the [RPC_V2_UREPLY_PRINTER](#) structure that **MUST** contain available notification data that matched the set of notifications that the client previously requested.

Return value/code	Description
0x00000000 ERROR_SUCCESS	This method MUST return zero to indicate successful completion or a nonzero Windows error code to indicate failure, as specified in [MS-ERREF] .

Upon receiving this message, the client **MUST** validate parameters as follows:

- Verify that the **hNotify** parameter is an RPC context handle to a notification object that was opened using **RpcReplyOpenPrinter**, and if that verification fails, return **ERROR_INVALID_HANDLE**, as specified in [\[MS-ERREF\]](#).
- Verify that the value of the **dwColor** parameter matches the last value that was passed in the **dwColor** parameter in the call to **RpcRouterRefreshPrinterChangeNotification**; if that

verification fails, set the **PRINTER_NOTIFY_INFO_COLORMISMATCH** bit in the variable pointed to by **pdwResult** and return 0.

If parameter validation fails, the client MUST fail the operation immediately and return a nonzero error response to the server. [<303>](#303)

Otherwise, the client MUST process the message as follows:

- Capture the **fdwFlags** in the notification state it maintains.
- Capture the notification data provided in the **Reply** parameter in the notification state.

3.2.4.2 Client Interaction with the Print Server

This section contains sequence specifications that the client MUST follow to perform specific tasks on the print server.

3.2.4.2.1 Printing a Document Using **RpcStartDocPrinter**

To print a document using [**RpcStartDocPrinter**](#RpcStartDocPrinter), the client MUST perform the following steps:

1. Invoke [**RpcOpenPrinter**](#RpcOpenPrinter), supplying the name of the target printer in the **pPrinterName** parameter and an **AccessRequired** value that includes [**PRINTER_ACCESS_USE**](#PRINTER_ACCESS_USE).
2. Using the printer handle obtained from **RpcOpenPrinter**, the client:
 1. MUST invoke **RpcStartDocPrinter** to initiate the print job.
 2. For each page in the print job, the client:
 1. SHOULD invoke [**RpcStartPagePrinter**](#RpcStartPagePrinter) to begin the page.
 2. MUST invoke [**RpcWritePrinter**](#RpcWritePrinter) to send the client's print data to the printer.
 3. SHOULD invoke [**RpcEndPagePrinter**](#RpcEndPagePrinter) to end the page.
 3. SHOULD invoke [**RpcEndDocPrinter**](#RpcEndDocPrinter) to end the print job.
3. If the client called **RpcEndDocPrinter** in step 2.3., the client MAY repeat from step 2 if there are additional print jobs to be sent to the printer.
4. If the client called **RpcEndDocPrinter** in step 2.3, the client SHOULD invoke [**RpcClosePrinter**](#RpcClosePrinter); otherwise, the client MUST invoke **RpcClosePrinter**, and the server MUST process the call under the assumption of an implicit call to **RpcEndDocPrinter**.

3.2.4.2.2 Printing a Document Using **RpcAddJob**

To print a document using [**RpcAddJob**](#RpcAddJob), the client MUST perform the following steps: [<304>](#304)

1. Invoke [**RpcOpenPrinter**](#RpcOpenPrinter), supplying the name of the target printer in the **pPrinterName** parameter and an **AccessRequired** value that includes [**PRINTER_ACCESS_USE**](#PRINTER_ACCESS_USE).
2. Using the printer handle that was obtained from **RpcOpenPrinter**, the client:
 1. MUST invoke **RpcAddJob**, supplying one for the **Level** parameter and an uninitialized [**ADDJOB_INFO_1**](#ADDJOB_INFO_1) structure, or supplying two or three for the **Level** parameter and an **ADDJOB_INFO_1** structure containing the client name.

2. Using the value that was returned in the **Path** member of the ADDJOB_INFO_1 structure for the target filename, the client MUST open the file, and MUST copy the data to be printed into it, and MUST close the file.
3. The client SHOULD invoke [RpcScheduleJob](#), supplying the value returned in the **JobId** member of the ADDJOB_INFO_1 structure for the **JobId** parameter.
3. The client MAY repeat step 2 if there are additional print jobs to send to the printer.
4. If the client called **RpcScheduleJob** in step 2.3, the client SHOULD invoke [RpcClosePrinter](#) with the printer handle obtained from **RpcOpenPrinter**; otherwise, the client MUST invoke **RpcClosePrinter**, and the server MUST process the call under the assumption of an implicit **RpcScheduleJob** for all jobs that have not been scheduled yet.

3.2.4.2.3 Enumerating Printers on a Print Server

To enumerate the printers on a print server, the client MUST perform the following steps:

1. Invoke [RpcEnumPrinters](#), supplying the name of the target print server in the **Name** parameter, the types of printers to enumerate in the **Flags** parameter, an information level value in **Level**, zero in **cbBuf**, and a pointer to a variable to store the required buffer size in **pcbNeeded**.
2. While **RpcEnumPrinters** returns with [ERROR_INSUFFICIENT_BUFFER](#), the client MUST:
 - Allocate new printer information buffer space with a size from the returned value for **pcbNeeded**.
 - Invoke **RpcEnumPrinters**, supplying the name of the target print server in the **Name** parameter, the types of printers to enumerate in the **Flags** parameter, an information level value in **Level**, a pointer to the printer information buffer in **pPrinterEnum**, the allocated size of the printer information buffer in **cbBuf**, a pointer to a variable to store the required buffer size in **pcbNeeded**, and a location to store the number of printer information items returned in **pcReturned**.

Note The "While" in the above description makes sense, considering that the number of printers MAY change at any time, and, therefore, the client SHOULD be prepared to receive **ERROR_INSUFFICIENT_BUFFER** even after allocating the correct buffer size the first time.

3.2.4.2.4 Enumerating Jobs on a Printer

To enumerate the jobs that are currently queued to a printer, the client MUST perform the following steps:

1. Invoke [RpcOpenPrinter](#), supplying the name of the target printer in the **pPrinterName** parameter, and an AccessRequired value that includes PRINTER_ACCESS_USE.
2. Using the printer handle that was obtained from **RpcOpenPrinter**, the client MUST:
 1. Set a local job position context to the desired starting index, typically zero.
 2. Set a local number of jobs to return in a single operation. [<305>](#)
 3. Until an [RpcEnumJobs](#) call returns with a success status and a **pcReturned** pointing to a value of zero, or until the expected set of jobs has been returned, the client MUST:

1. Invoke **RpcEnumJobs**, supplying the job position context in **FirstJob**, the number of jobs to return in a call in **NoJobs**, the desired information level in **Level**, a pointer to the job information buffer in **pJob**, the size of the job information buffer in **cbBuf**, a pointer to a variable to store the required buffer size in **pcbNeeded**, and a pointer to a variable to store the number of job information structures returned in **pcReturned**.
2. While **RpcEnumJobs** returns with **ERROR_INSUFFICIENT_BUFFER**, as specified in [\[MS-ERREF\]](#), the client MUST:
 1. Allocate a new job information buffer with the size returned in **pcbNeeded**.
 2. Invoke **RpcEnumJobs**, supplying the job position context in **FirstJob**, the number of jobs to return in a call in **NoJobs**, the desired information level in **Level**, a pointer to the job information buffer in **pJob**, the size of the job information buffer in **cbBuf**, a pointer to a variable to store the required buffer size in **pcbNeeded**, and a pointer to a variable to store the number of job information structures in **pcReturned**.

Note The "While" in the above description makes sense, considering that the number of jobs MAY change at any time, and therefore, the client SHOULD be prepared to receive **ERROR_INSUFFICIENT_BUFFER**, as specified in [\[MS-ERREF\]](#), even after allocating the correct buffer size the first time.

3. Increase the local job position context by the value supplied in **pcReturned**.
3. The client SHOULD invoke [RpcClosePrinter](#) with the printer handle obtained from **RpcOpenPrinter**, or SHOULD repeat step 2 if there are further job enumeration requests to make.

3.2.4.2.5 Receiving Notifications from a Print Server

To receive notifications for a printing event, a client MUST perform the following steps:

- Invoke [RpcOpenPrinter](#), supplying the name of the target printer in the **pPrinterName** parameter and an **AccessRequired** value that includes [PRINTER_ACCESS_USE](#).
- Using the printer handle that was obtained from **RpcOpenPrinter**, the client MUST invoke [RpcRemoteFindFirstPrinterChangeNotificationEx](#), supplying the notification flags, and the job and printer fields that notifications are to be delivered for. The call MUST also supply a value in **dwPrinterLocal** that the client will use to identify the source for the later notifications.

The server opens a channel to the client as a result of processing this call by calling the client's [RpcReplyOpenPrinter](#) method. Therefore, the client MUST implement the Print System Remote Protocol server interface to process notifications.

- The client MUST process notifications as follows:
 - Process an **RpcReplyOpenPrinter** call, using the value in **dwPrinterRemote** to determine the client context established by the **dwPrinterLocal** parameter in a previous **RpcRemoteFindFirstPrinterChangeNotificationEx** call.
 - This call MUST produce an **RPC** handle that the later notifications use.
 - Process [RpcRouterReplyPrinterEx](#) method calls. The values for **fdwFlags**, **cbBuffer**, and **pBuffer** MUST specify the notification type and any associated data.

- To terminate the notifications, the client SHOULD invoke [RpcFindClosePrinterChangeNotification](#), supplying the printer handle obtained from **RpcOpenPrinter**.

The server closes the channel to the client as a result of processing this call by calling [RpcReplyClosePrinter](#) on the client.

- The client MUST process an **RpcReplyClosePrinter** to terminate the notification sequence. This provides the **RPC** handle from the associated **RpcReplyOpenPrinter** call.
- The client SHOULD call [RpcClosePrinter](#). If the client has not called **RpcFindClosePrinterChangeNotification** in step 4, the server MUST implicitly close the notification channel, and in the processing of that, call **RpcReplyClosePrinter**, which MUST be processed by the client as specified in step 5.

3.2.4.2.6 Announcing Shared Printers to Print Servers

To announce its shared printers to print servers, the client MUST perform these steps:

1. Make a policy-specific determination whether shared printers should be enumerated to print servers. [<306>](#)
2. If shared printers should be enumerated, for each printer installed on the client that has the [PRINTER_ATTRIBUTE_SHARED](#) set, create a [PRINTER_CONTAINER](#) with Level set to one, and populate it with a [PRINTER_INFO_1](#) describing the printer, and then call the print server's [RpcAddPrinter](#) or [RpcAddPrinterEx](#) method. [<307>](#)

3.2.4.2.7 Adding a Printer to a Print Server

To add a printer to a print server, the client performs these steps:

1. The client MAY use methods defined by this protocol to query the print server for information used to initialize other data structures. [<308>](#)
2. The client SHOULD call the print server's [RpcEnumPrinterDrivers](#) to determine whether a printer driver for the new printer is already installed on the server.
3. If a printer driver is not already installed, the client SHOULD call [RpcAddPrinterDriver](#) or [RpcAddPrinterDriverEx](#) to install a printer driver for the new printer.
4. The client MUST allocate a [PRINTER_CONTAINER](#) structure and populate it with a [PRINTER_INFO_2](#) structure describing the new printer.
5. The client MUST allocate a [DEVMODE_CONTAINER](#) and populate it with the default DEVMODE for the new printer.
6. The client MUST allocate a [SECURITY_CONTAINER](#) and populate it with a [SECURITY_DESCRIPTOR](#) containing the security information for the new printer.
7. The client MUST call the print server's [RpcAddPrinter](#) with the print server's name, and the CONTAINER parameters from steps 4, 5, and 6. Alternatively, the client can use the [RpcAddPrinterEx](#) and specify an additional [SPLCLIENT_CONTAINER](#) that describes the client in more detail. **RpcAddPrinterEx** returns a **PRINTER_HANDLE** to the newly added printer in the variable pointed to by **pHandle**. The client SHOULD close that handle using [RpcClosePrinter](#) when it no longer needs it.

3.2.5 Timer Events

No protocol timer events are required on the client beyond the timers required in the underlying RPC protocol.

3.2.6 Other Local Events

A client's invocation of each method is typically the result of local application activity. The local application on the client computer specifies values for all input parameters. No other higher-layer triggered events are processed. The values specified for input parameters are described in [section 2](#).

No additional local events are used on the client beyond the events maintained in the underlying RPC protocol.

4 Protocol Examples

The following sections describe several operations as used in common scenarios to illustrate the function of the Print System Remote Protocol.

Each subsection shows a sequence diagram including the necessary steps for a common scenario.

4.1 Adding a Printer to a Server

To add a printer to a print server, a client SHOULD perform the following steps as shown in the figure below:

1. Enumerate existing printer drivers using [RpcEnumPrinterDrivers](#).
2. Select an existing printer driver or add a new printer driver using [RpcAddPrinterDriver](#).
3. Populate a [PRINTER_INFO_2](#) structure with information about the new printer, and call [RpcAddPrinter](#).
4. Close the returned [PRINTER_HANDLE](#) using [RpcClosePrinter](#).

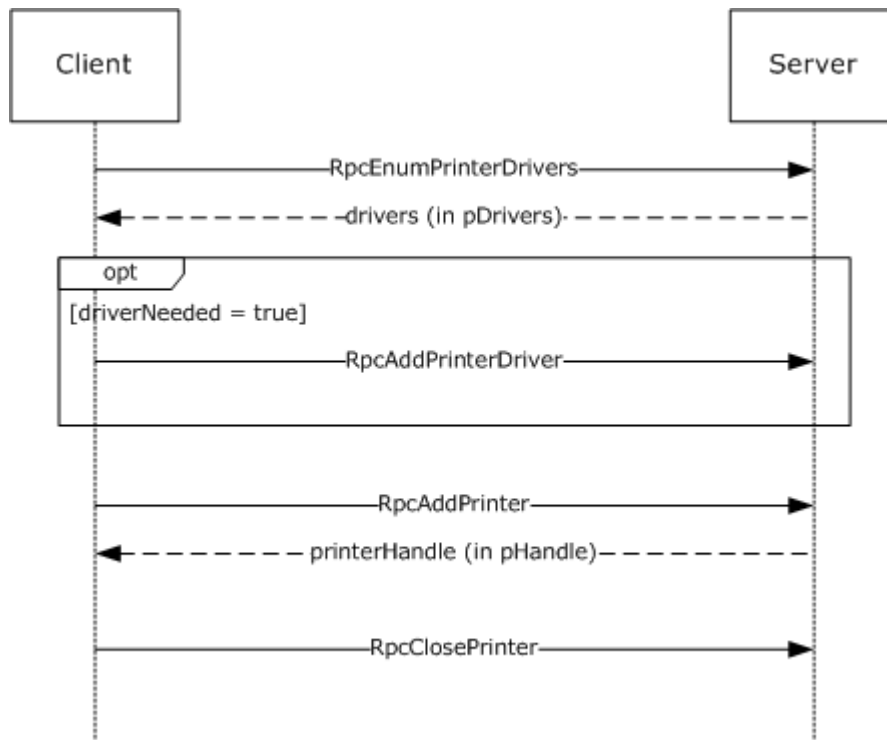


Figure 4: Adding a printer to a server

4.2 Adding a Printer Driver to a Server

To add a printer driver to a print server, a client SHOULD perform the following steps as shown in the figure below:

1. Enumerate existing printer drivers using [RpcEnumPrinterDrivers](#).

2. If the driver does not already exist, use [RpcAddPrinterDriver](#) to add the driver to the print server.

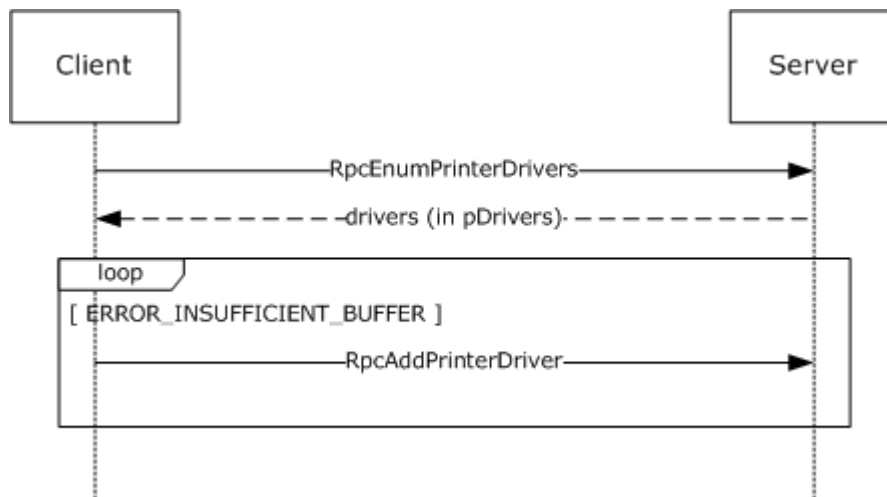


Figure 5: Adding a printer driver to a server

4.3 Enumerating and Managing Printers

In order to manage printers on a print server, a client SHOULD perform the following steps as shown in the figure below:

1. Enumerate existing printers using [RpcEnumPrinters](#).
2. Retrieve current information about a printer using [RpcGetPrinter](#).
3. Use [RpcSetPrinter](#) to modify the state of the printer.

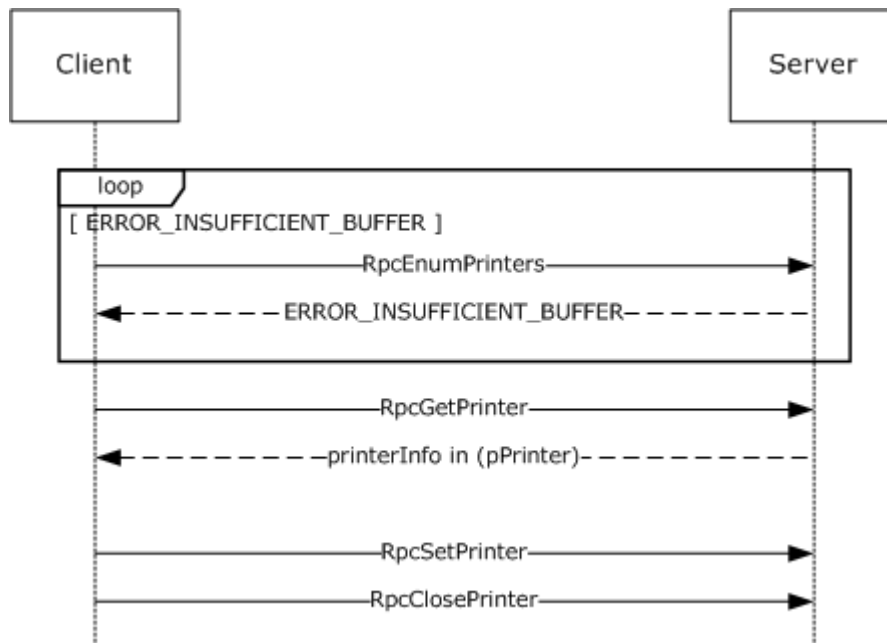


Figure 6: Enumerating and managing printers on a server

4.4 Printing a Job, Job Scheduling, and Setting Job Information

With servers supporting [RpcAddPrinter](#), a client MAY use the following steps to submit a print job as shown in the figure below:

1. Open a **PRINTER_HANDLE** using [RpcOpenPrinter](#).
2. Use [RpcAddJob](#) to add a new job to the print server.
3. Write the print data to the file identified in the returned [ADDJOB_INFO_1](#) structure.
4. Schedule the job for printing using [RpcScheduleJob](#).
5. Optionally modify the job status or job priority using [RpcSetJob](#).
6. Close the **PRINTER_HANDLE** using [RpcClosePrinter](#).

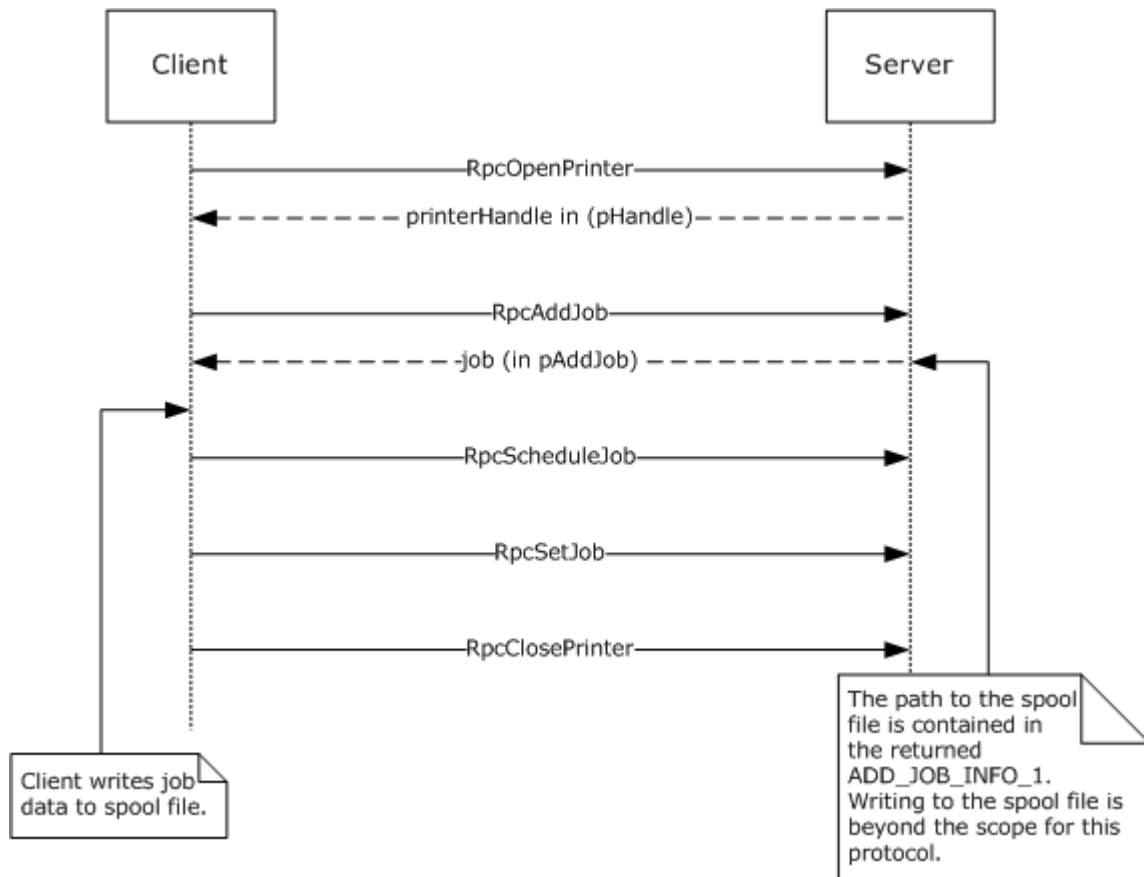


Figure 7: Printing a job, job scheduling, and setting job information

4.5 Enumerating Jobs and Modifying Job Settings

In order to enumerate print jobs on a server, modify job settings, or change job priorities, the client SHOULD perform the following steps as shown in the figure below:

1. Open the printer using [RpcOpenPrinter](#).
2. Enumerate jobs scheduled for printing on the printer using [RpcEnumJobs](#).
3. Modify job settings or job priority using [RpcSetJob](#).
4. Close the printer using [RpcClosePrinter](#).

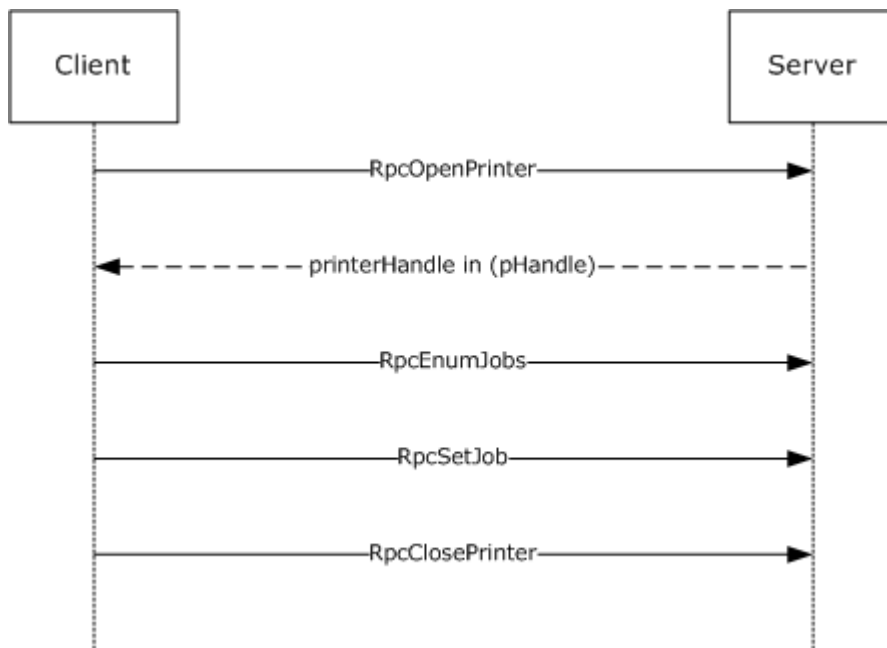


Figure 8: Enumerating jobs and modifying job settings

4.6 Receiving Notifications on Printing Events

In order to receive notifications about state changes of print servers, printers and print jobs, the client SHOULD perform the following steps as shown in the figure below:

1. Open the print server or printer using [RpcOpenPrinter](#).
2. Register for change notifications using [RpcRemoteFindFirstPrinterChangeNotificationEx](#):
 - The server will call the client's [RpcReplyOpenPrinter](#) method to open a reverse channel, which will be used to send change notifications to the client. The client MUST return an RPC binding handle identifying the reverse channel.
 - As long as the client stays registered for notifications, the server will call the client's [RpcRouterReplyPrinterEx](#) method for each change of the requested type occurring on the server.
3. The client MAY route the change notifications to applications or process it otherwise to reflect state changes.
4. When the client is no longer interested in state changes, it SHOULD unregister from notifications by calling [RpcFindClosePrinterChangeNotification](#) with the handle returned by the initial call to [RpcRemoteFindFirstPrinterChangeNotificationEx](#).

- The server will call the client's [RpcReplyClosePrinter](#) with the handle previously obtained by **RpcReplyOpenPrinter** to notify the client that the binding handle for the reverse channel SHOULD be closed.

5. The client SHOULD close the print server or [PRINTER_HANDLE](#) using [RpcClosePrinter](#).

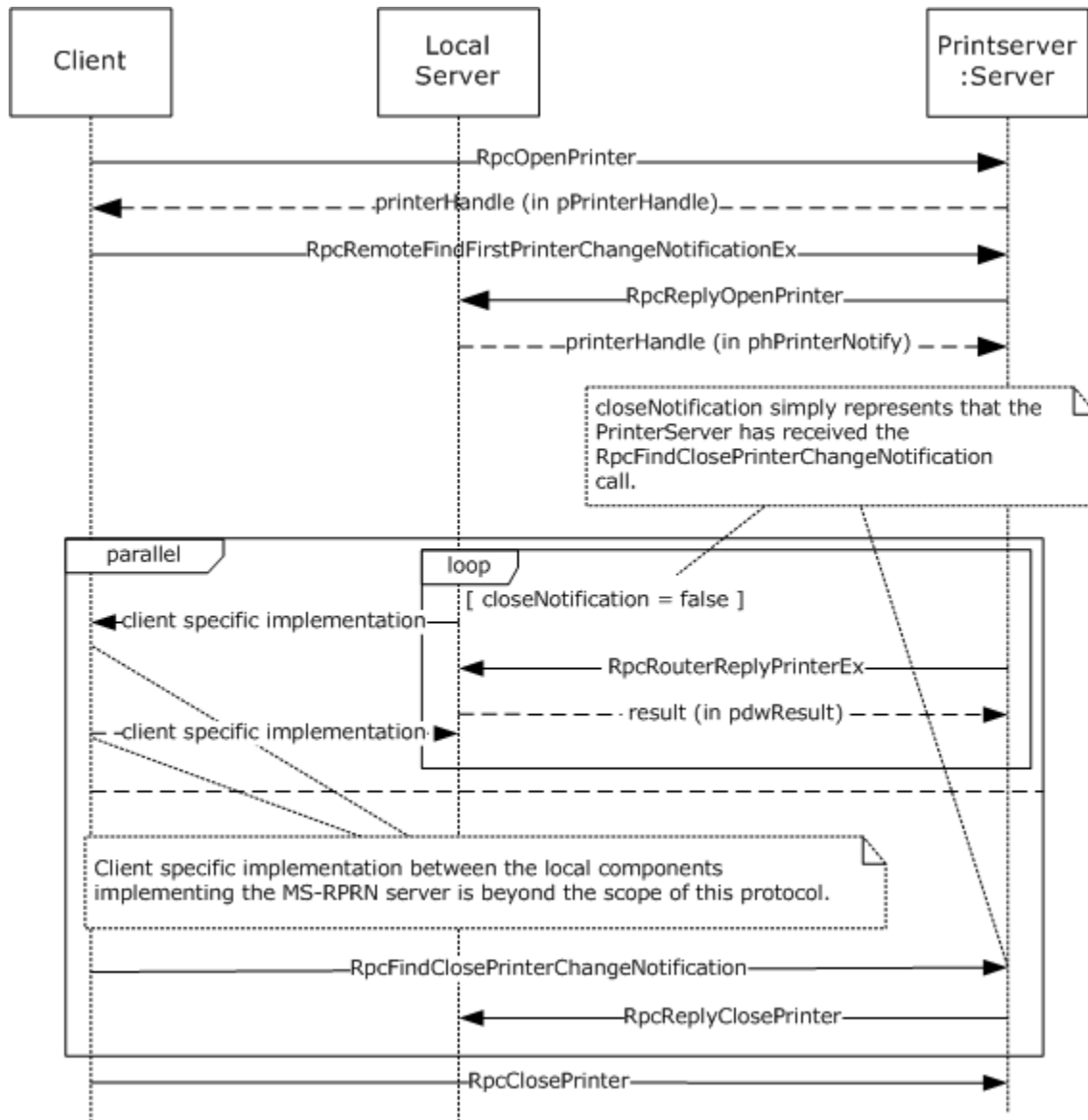


Figure 9: Receiving notifications on different printing events

5 Security

The following sections specify security considerations for implementers of the Print System Remote Protocol.

5.1 Security Considerations for Implementers

Does not apply to this protocol.

5.2 Index of Security Parameters

Security considerations for both unauthenticated and authenticated RPC are specified in [\[C706\]](#) (sections [2.3](#) and [13](#)).<309>

6 Appendix A: Full IDL

For ease of implementation, the full stand-alone Interface Definition Language (IDL) is provided. Some of the data types and structures used by this protocol are defined in other documents. In order for this IDL to stand alone, those types and structures, from [\[MS-DTYP\]](#), are included below.

```
import "ms-dtyp.idl";

#if __midl < 700
#define disable_consistency_check
#endif

// [MS-RPRN] common constants
#define TABLE_DWORD          0x1
#define TABLE_STRING         0x2
#define TABLE_DEVMODE        0x3
#define TABLE_TIME           0x4
#define TABLE_SECURITYDESCRIPTOR 0x5

#define CCHDEVICENAME          32
#define CCHFORMNAME            32

// [MS-RPRN] common enumerations
typedef enum {
    BIDI_NULL    = 0,
    BIDI_INT     = 1,
    BIDI_FLOAT   = 2,
    BIDI_BOOL    = 3,
    BIDI_STRING  = 4,
    BIDI_TEXT    = 5,
    BIDI_ENUM    = 6,
    BIDI_BLOB    = 7
} BIDI_TYPE;

// [MS-RPRN] common data types
typedef unsigned short LANGID;
typedef [context_handle] void* GDI_HANDLE;
typedef [context_handle] void* PRINTER_HANDLE;
typedef [handle] wchar_t* STRING_HANDLE;

// [MS-RPRN] common utility structures
typedef struct {
    long cx;
    long cy;
} SIZE;

typedef struct {
    long left;
    long top;
    long right;
    long bottom;
} RECTL;

// [MS-RPRN] common device state structure
typedef struct _devicemode {
    wchar_t  dmDeviceName[CCHDEVICENAME];
```

```

    unsigned short dmSpecVersion;
    unsigned short dmDriverVersion;
    unsigned short dmSize;
    unsigned short dmDriverExtra;

    DWORD dmFields;

    short dmOrientation;
    short dmPaperSize;
    short dmPaperLength;
    short dmPaperWidth;
    short dmScale;
    short dmCopies;
    short dmDefaultSource;
    short dmPrintQuality;
    short dmColor;
    short dmDuplex;
    short dmYResolution;
    short dmTTOption;
    short dmCollate;

    wchar_t dmFormName[CCHFORMNAME];

    unsigned short reserved0;

    DWORD reserved1;
    DWORD reserved2;
    DWORD reserved3;
    DWORD dmNup;
    DWORD reserved4;
    DWORD dmICMMethod;
    DWORD dmICMIntent;
    DWORD dmMediaType;
    DWORD dmDitherType;
    DWORD reserved5;
    DWORD reserved6;
    DWORD reserved7;
    DWORD reserved8;
} DEVMODE;

// [MS-RPRN] common info structures
typedef struct _DOC_INFO_1 {
    [string] wchar_t* pDocName;
    [string] wchar_t* pOutputFile;
    [string] wchar_t* pData-type;
} DOC_INFO_1;

typedef struct _DRIVER_INFO_1 {
    [string] wchar_t* pName;
} DRIVER_INFO_1;

typedef struct _DRIVER_INFO_2 {
    DWORD cVersion;
    [string] wchar_t* pName;
    [string] wchar_t* pEnvironment;
    [string] wchar_t* pDriverPath;
    [string] wchar_t* pDataFile;

```

```

    [string] wchar_t* pConfigFile;
} DRIVER_INFO_2;

typedef struct _RPC_DRIVER_INFO_3 {
    DWORD cVersion;
    [string] wchar_t* pName;
    [string] wchar_t* pEnvironment;
    [string] wchar_t* pDriverPath;
    [string] wchar_t* pDataFile;
    [string] wchar_t* pConfigFile;
    [string] wchar_t* pHelpFile;
    [string] wchar_t* pMonitorName;
    [string] wchar_t* pDefaultDataType;
    DWORD cchDependentFiles;
    [size_is(cchDependentFiles), unique]
    wchar_t* pDependentFiles;
} RPC_DRIVER_INFO_3;

typedef struct _RPC_DRIVER_INFO_4 {
    DWORD cVersion;
    [string] wchar_t* pName;
    [string] wchar_t* pEnvironment;
    [string] wchar_t* pDriverPath;
    [string] wchar_t* pDataFile;
    [string] wchar_t* pConfigFile;
    [string] wchar_t* pHelpFile;
    [string] wchar_t* pMonitorName;
    [string] wchar_t* pDefaultDataType;
    DWORD cchDependentFiles;
    [size_is(cchDependentFiles), unique]
    wchar_t* pDependentFiles;
    DWORD cchPreviousNames;
    [size_is(cchPreviousNames), unique]
    wchar_t* pszzPreviousNames;
} RPC_DRIVER_INFO_4;

typedef struct _RPC_DRIVER_INFO_6 {
    DWORD cVersion;
    [string] wchar_t* pName;
    [string] wchar_t* pEnvironment;
    [string] wchar_t* pDriverPath;
    [string] wchar_t* pDataFile;
    [string] wchar_t* pConfigFile;
    [string] wchar_t* pHelpFile;
    [string] wchar_t* pMonitorName;
    [string] wchar_t* pDefaultDataType;
    DWORD cchDependentFiles;
    [size_is(cchDependentFiles), unique]
    wchar_t* pDependentFiles;
    DWORD cchPreviousNames;
    [size_is(cchPreviousNames), unique]
    wchar_t* pszzPreviousNames;
    FILETIME ftDriverDate;
    DWORDLONG dwlDriverVersion;
    [string] wchar_t* pMfgName;
    [string] wchar_t* pOEMUrl;
    [string] wchar_t* pHardwareID;
    [string] wchar_t* pProvider;

```

```

} RPC_DRIVER_INFO_6;

typedef struct _RPC_DRIVER_INFO_8 {
    DWORD cVersion;
    [string] wchar_t* pName;
    [string] wchar_t* pEnvironment;
    [string] wchar_t* pDriverPath;
    [string] wchar_t* pDataFile;
    [string] wchar_t* pConfigFile;
    [string] wchar_t* pHelpFile;
    [string] wchar_t* pMonitorName;
    [string] wchar_t* pDefaultDataType;
    DWORD cchDependentFiles;
    [size_is(cchDependentFiles), unique]
    wchar_t* pDependentFiles;
    DWORD cchPreviousNames;
    [size_is(cchPreviousNames), unique]
    wchar_t* pszzPreviousNames;
    FILETIME ftDriverDate;
    DWORDLONG dwlDriverVersion;
    [string] wchar_t* pMfgName;
    [string] wchar_t* pOEMUrl;
    [string] wchar_t* pHardwareID;
    [string] wchar_t* pProvider;
    [string] wchar_t* pPrintProcessor;
    [string] wchar_t* pVendorSetup;
    DWORD cchColorProfiles;
    [size_is(cchColorProfiles), unique]
    wchar_t* pszzColorProfiles;
    [string] wchar_t* pInfPath;
    DWORD dwPrinterDriverAttributes;
    DWORD cchCoreDependencies;
    [size_is(cchCoreDependencies), unique]
    wchar_t* pszzCoreDriverDependencies;
    FILETIME ftMinInboxDriverVerDate;
    DWORDLONG dwlMinInboxDriverVerVersion;
} RPC_DRIVER_INFO_8;

typedef struct _FORM_INFO_1 {
    DWORD Flags;
    [string] wchar_t* pName;
    SIZE Size;
    RECTL ImageableArea;
} FORM_INFO_1;

typedef struct _RPC_FORM_INFO_2 {
    DWORD Flags;
    [string, unique] const wchar_t* pName;
    SIZE Size;
    RECTL ImageableArea;
    [string, unique] const char* pKeyword;
    DWORD StringType;
    [string, unique] const wchar_t* pMuiDll;
    DWORD dwResourceId;
    [string, unique] const wchar_t* pDisplayName;
    LANGID wLangID;
} RPC_FORM_INFO_2;

```

```

typedef struct _JOB_INFO_1 {
    DWORD JobId;
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pMachineName;
    [string] wchar_t* pUserName;
    [string] wchar_t* pDocument;
    [string] wchar_t* pDataatype;
    [string] wchar_t* pStatus;
    DWORD Status;
    DWORD Priority;
    DWORD Position;
    DWORD TotalPages;
    DWORD PagesPrinted;
    SYSTEMTIME Submitted;
} JOB_INFO_1;

typedef struct _JOB_INFO_2 {
    DWORD JobId;
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pMachineName;
    [string] wchar_t* pUserName;
    [string] wchar_t* pDocument;
    [string] wchar_t* pNotifyName;
    [string] wchar_t* pDataatype;
    [string] wchar_t* pPrintProcessor;
    [string] wchar_t* pParameters;
    [string] wchar_t* pDriverName;
    DEVMODE* pDevMode;
    [string] wchar_t* pStatus;
    SECURITY_DESCRIPTOR* pSecurityDescriptor;
    DWORD Status;
    DWORD Priority;
    DWORD Position;
    DWORD StartTime;
    DWORD UntilTime;
    DWORD TotalPages;
    DWORD Size;
    SYSTEMTIME Submitted;
    DWORD Time;
    DWORD PagesPrinted;
} JOB_INFO_2;

typedef struct _JOB_INFO_3 {
    DWORD JobId;
    DWORD NextJobId;
    DWORD Reserved;
} JOB_INFO_3;

typedef struct _JOB_INFO_4 {
    DWORD JobId;
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pMachineName;
    [string] wchar_t* pUserName;
    [string] wchar_t* pDocument;
    [string] wchar_t* pNotifyName;
    [string] wchar_t* pDataatype;
    [string] wchar_t* pPrintProcessor;
    [string] wchar_t* pParameters;
}

```

```

    [string] wchar_t* pDriverName;
    DEVMODE* pDevMode;
    [string] wchar_t* pStatus;
    SECURITY_DESCRIPTOR* pSecurityDescriptor;
    DWORD Status;
    DWORD Priority;
    DWORD Position;
    DWORD StartTime;
    DWORD UntilTime;
    DWORD TotalPages;
    DWORD Size;
    SYSTEMTIME Submitted;
    DWORD Time;
    DWORD PagesPrinted;
    long SizeHigh;
} JOB_INFO_4;

typedef struct _MONITOR_INFO_1 {
    [string] wchar_t* pName;
} MONITOR_INFO_1;

typedef struct _MONITOR_INFO_2 {
    [string] wchar_t* pName;
    [string] wchar_t* pEnvironment;
    [string] wchar_t* pDLLName;
} MONITOR_INFO_2;

typedef struct _PORT_INFO_1 {
    [string] wchar_t* pName;
} PORT_INFO_1;

typedef struct _PORT_INFO_2 {
    [string] wchar_t* pPortName;
    [string] wchar_t* pMonitorName;
    [string] wchar_t* pDescription;
    DWORD fPortType;
    DWORD Reserved;
} PORT_INFO_2;

typedef struct _PORT_INFO_3 {
    DWORD dwStatus;
    [string] wchar_t* pszStatus;
    DWORD dwSeverity;
} PORT_INFO_3;

typedef struct _PORT_INFO_FF {
    wchar_t* pName;
    DWORD cbMonitorData;
    BYTE* pMonitorData;
} PORT_INFO_FF;

typedef struct _PRINTER_INFO_STRESS {
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pServerName;
    DWORD cJobs;
    DWORD cTotalJobs;
    DWORD cTotalBytes;
    SYSTEMTIME stUpTime;
}

```

```

    DWORD MaxcRef;
    DWORD cTotalPagesPrinted;
    DWORD dwGetVersion;
    DWORD fFreeBuild;
    DWORD cSpooling;
    DWORD cMaxSpooling;
    DWORD cRef;
    DWORD cErrorOutOfPaper;
    DWORD cErrorNotReady;
    DWORD cJobError;
    DWORD dwNumberOfProcessors;
    DWORD dwProcessorType;
    DWORD dwHighPartTotalBytes;
    DWORD cChangeID;
    DWORD dwLastError;
    DWORD Status;
    DWORD cEnumerateNetworkPrinters;
    DWORD cAddNetPrinters;
    unsigned short wProcessorArchitecture;
    unsigned short wProcessorLevel;
    DWORD cRefIC;
    DWORD dwReserved2;
    DWORD dwReserved3;
} PRINTER_INFO_STRESS;

typedef struct _PRINTER_INFO_1 {
    DWORD Flags;
    [string] wchar_t* pDescription;
    [string] wchar_t* pName;
    [string] wchar_t* pComment;
} PRINTER_INFO_1;

typedef struct _PRINTER_INFO_2 {
    [string] wchar_t* pServerName;
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pShareName;
    [string] wchar_t* pPortName;
    [string] wchar_t* pDriverName;
    [string] wchar_t* pComment;
    [string] wchar_t* pLocation;
    DEVMODE* pDevMode;
    [string] wchar_t* pSepFile;
    [string] wchar_t* pPrintProcessor;
    [string] wchar_t* pDataatype;
    [string] wchar_t* pParameters;
    SECURITY_DESCRIPTOR* pSecurityDescriptor;
    DWORD Attributes;
    DWORD Priority;
    DWORD DefaultPriority;
    DWORD StartTime;
    DWORD UntilTime;
    DWORD Status;
    DWORD cJobs;
    DWORD AveragePPM;
} PRINTER_INFO_2;

typedef struct _PRINTER_INFO_3 {
    SECURITY_DESCRIPTOR* pSecurityDescriptor;

```

```

} PRINTER_INFO_3;

typedef struct _PRINTER_INFO_4 {
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pServerName;
    DWORD Attributes;
} PRINTER_INFO_4;

typedef struct _PRINTER_INFO_5 {
    [string] wchar_t* pPrinterName;
    [string] wchar_t* pPortName;
    DWORD Attributes;
    DWORD DeviceNotSelectedTimeout;
    DWORD TransmissionRetryTimeout;
} PRINTER_INFO_5;

typedef struct _PRINTER_INFO_6 {
    DWORD dwStatus;
} PRINTER_INFO_6;

typedef struct _PRINTER_INFO_7 {
    [string] wchar_t* pszObjectGUID;
    DWORD dwAction;
} PRINTER_INFO_7;

typedef struct _PRINTER_INFO_8 {
    DEVMODE* pDevMode;
} PRINTER_INFO_8;

typedef struct _PRINTER_INFO_9 {
    DEVMODE* pDevMode;
} PRINTER_INFO_9;

typedef struct _SPLCLIENT_INFO_1 {
    DWORD dwSize;
    wchar_t* pMachineName;
    wchar_t* pUserName;
    DWORD dwBuildNum;
    DWORD dwMajorVersion;
    DWORD dwMinorVersion;
    unsigned short wProcessorArchitecture;
} SPLCLIENT_INFO_1;

typedef struct _SPLCLIENT_INFO_3 {
    unsigned int cbSize;
    DWORD dwFlags;
    DWORD dwSize;
    [string] wchar_t* pMachineName;
    [string] wchar_t* pUserName;
    DWORD dwBuildNum;
    DWORD dwMajorVersion;
    DWORD dwMinorVersion;
    unsigned short wProcessorArchitecture;
    unsigned __int64 hSplPrinter;
} SPLCLIENT_INFO_3;

// [MS-RPRN] common info container structures
typedef struct _DEVMODE_CONTAINER {

```

```

        DWORD cbBuf;
        [size_is(cbBuf), unique] BYTE* pDevMode;
    } DEVMODE_CONTAINER;

typedef struct _DOC_INFO_CONTAINER {
    DWORD Level;
    [switch_is(Level)] union {
        [case(1)]
            DOC_INFO_1* pDocInfo1;
    } DocInfo;
} DOC_INFO_CONTAINER;

typedef struct _DRIVER_CONTAINER {
    DWORD Level;
    [switch_is(Level)] union {
        [case(1)]
            DRIVER_INFO_1* Level1;
        [case(2)]
            DRIVER_INFO_2* Level2;
        [case(3)]
            RPC_DRIVER_INFO_3* Level3;
        [case(4)]
            RPC_DRIVER_INFO_4* Level4;
        [case(6)]
            RPC_DRIVER_INFO_6* Level6;
        [case(8)]
            RPC_DRIVER_INFO_8* Level8;
    } DriverInfo;
} DRIVER_CONTAINER;

typedef struct _FORM_CONTAINER {
    DWORD Level;
    [switch_is(Level)] union {
        [case(1)]
            FORM_INFO_1* pFormInfo1;
        [case(2)]
            RPC_FORM_INFO_2* pFormInfo2;
    } FormInfo;
} FORM_CONTAINER;

typedef struct _JOB_CONTAINER {
    DWORD Level;
    [switch_is(Level)] union {
        [case(1)]
            JOB_INFO_1* Level1;
        [case(2)]
            JOB_INFO_2* Level2;
        [case(3)]
            JOB_INFO_3* Level3;
        [case(4)]
            JOB_INFO_4* Level4;
    } JobInfo;
} JOB_CONTAINER;

typedef struct _MONITOR_CONTAINER {
    DWORD Level;
    [switch_is(Level)] union {
        [case(1)]

```

```

        MONITOR_INFO_1* pMonitorInfo1;
    [case(2)]
        MONITOR_INFO_2* pMonitorInfo2;
    } MonitorInfo;
} MONITOR_CONTAINER;

typedef struct _PORT_CONTAINER {
    DWORD Level;
    [switch_is(0x00FFFFFF & Level)]
    union {
        [case(1)]
            PORT_INFO_1* pPortInfo1;
        [case(2)]
            PORT_INFO_2* pPortInfo2;
        [case(3)]
            PORT_INFO_3* pPortInfo3;
        [case(0x00FFFFFF)]
            PORT_INFO_FF* pPortInfoFF;
    } PortInfo;
} PORT_CONTAINER;

typedef struct _PORT_VAR_CONTAINER {
    DWORD cbMonitorData;
    [size_is(cbMonitorData), unique, disable_consistency_check] BYTE*
        pMonitorData;
} PORT_VAR_CONTAINER;

typedef struct _PRINTER_CONTAINER {
    DWORD Level;
    [switch_is(Level)] union {
        [case(0)]
            PRINTER_INFO_STRESS* pPrinterInfoStress;
        [case(1)]
            PRINTER_INFO_1* pPrinterInfo1;
        [case(2)]
            PRINTER_INFO_2* pPrinterInfo2;
        [case(3)]
            PRINTER_INFO_3* pPrinterInfo3;
        [case(4)]
            PRINTER_INFO_4* pPrinterInfo4;
        [case(5)]
            PRINTER_INFO_5* pPrinterInfo5;
        [case(6)]
            PRINTER_INFO_6* pPrinterInfo6;
        [case(7)]
            PRINTER_INFO_7* pPrinterInfo7;
        [case(8)]
            PRINTER_INFO_8* pPrinterInfo8;
        [case(9)]
            PRINTER_INFO_9* pPrinterInfo9;
    } PrinterInfo;
} PRINTER_CONTAINER;

typedef struct _RPC_BINARY_CONTAINER {
    DWORD cbBuf;
    [size_is(cbBuf), unique] BYTE* pszString;
} RPC_BINARY_CONTAINER;

```

```

typedef struct _RPC_BIDI_DATA {
    DWORD dwBidiType;
    [switch_is(dwBidiType)] union {
        [case(BIDI_NULL, BIDI_BOOL)]
            int bData;
        [case(BIDI_INT)]
            long iData;
        [case(BIDI_STRING, BIDI_TEXT, BIDI_ENUM)]
            [string,unique] wchar_t* sData;
        [case(BIDI_FLOAT)]
            float fData;
        [case(BIDI_BLOB)]
            RPC_BINARY_CONTAINER biData;
    } u;
} RPC_BIDI_DATA;

typedef struct _RPC_BIDI_REQUEST_DATA {
    DWORD dwReqNumber;
    [string, unique] wchar_t* pSchema;
    RPC_BIDI_DATA data;
} RPC_BIDI_REQUEST_DATA;

typedef struct _RPC_BIDI_RESPONSE_DATA {
    DWORD dwResult;
    DWORD dwReqNumber;
    [string, unique] wchar_t* pSchema;
    RPC_BIDI_DATA data;
} RPC_BIDI_RESPONSE_DATA;

typedef struct _RPC_BIDI_REQUEST_CONTAINER {
    DWORD Version;
    DWORD Flags;
    DWORD Count;
    [size_is(Count), unique] RPC_BIDI_REQUEST_DATA aData[];
} RPC_BIDI_REQUEST_CONTAINER;

typedef struct _RPC_BIDI_RESPONSE_CONTAINER {
    DWORD Version;
    DWORD Flags;
    DWORD Count;
    [size_is(Count), unique] RPC_BIDI_RESPONSE_DATA aData[];
} RPC_BIDI_RESPONSE_CONTAINER;

typedef struct SECURITY_CONTAINER {
    DWORD cbBuf;
    [size_is(cbBuf), unique] BYTE* pSecurity;
} SECURITY_CONTAINER;

typedef struct _SPLCLIENT_CONTAINER {
    DWORD Level;
    [switch_is(Level)] union {
        [case(1)]
            SPLCLIENT_INFO_1* pClientInfo1;
        [case(2)]
            unsigned __int64 pClientInfo2;
        [case(3)]
            SPLCLIENT_INFO_3* pClientInfo3;
    } ClientInfo;
}

```

```

} SPLCLIENT_CONTAINER;

typedef struct _STRING_CONTAINER {
    DWORD cbBuf;
    [size_is(cbBuf/2), unique] WCHAR* pszString;
} STRING_CONTAINER;

typedef struct _SYSTEMTIME_CONTAINER {
    DWORD cbBuf;
    SYSTEMTIME* pSystemTime;
} SYSTEMTIME_CONTAINER;

typedef struct _RPC_V2_NOTIFY_OPTIONS_TYPE {
    unsigned short Type;
    unsigned short Reserved0;
    DWORD Reserved1;
    DWORD Reserved2;
    DWORD Count;
    [size_is(Count), unique] unsigned short* pFields;
} RPC_V2_NOTIFY_OPTIONS_TYPE;

typedef struct _RPC_V2_NOTIFY_OPTIONS {
    DWORD Version;
    DWORD Reserved;
    DWORD Count;
    [size_is(Count), unique] RPC_V2_NOTIFY_OPTIONS_TYPE* pTypes;
} RPC_V2_NOTIFY_OPTIONS;

typedef
[switch_type (DWORD)]
union _RPC_V2_NOTIFY_INFO_DATA_DATA {
    [case(TABLE_STRING)]
        STRING_CONTAINER String;
    [case(TABLE_DWORD)]
        DWORD dwData[2];
    [case(TABLE_TIME)]
        SYSTEMTIME_CONTAINER SystemTime;
    [case(TABLE_DEVMODE)]
        DEVMODE_CONTAINER DevMode;
    [case(TABLE_SECURITYDESCRIPTOR)]
        SECURITY_CONTAINER SecurityDescriptor;
} RPC_V2_NOTIFY_INFO_DATA_DATA;

typedef struct _RPC_V2_NOTIFY_INFO_DATA {
    unsigned short Type;
    unsigned short Field;
    DWORD Reserved;
    DWORD Id;
    [switch_is(Reserved & 0xffff)]
        RPC_V2_NOTIFY_INFO_DATA_DATA Data;
} RPC_V2_NOTIFY_INFO_DATA;

typedef struct _RPC_V2_NOTIFY_INFO {
    DWORD Version;
    DWORD Flags;
    DWORD Count;
    [size_is(Count), unique] RPC_V2_NOTIFY_INFO_DATA aData[];
} RPC_V2_NOTIFY_INFO;

```

```

typedef [switch_type(DWORD)] union _RPC_V2_UREPLY_PRINTER {
    [case (0)]
        RPC_V2_NOTIFY_INFO* pInfo;
} RPC_V2_UREPLY_PRINTER;

// [MS-RPRN] interface
[
    uuid(12345678-1234-ABCD-EF00-0123456789AB),
    version(1.0),
    ms_union,
    endpoint("ncacn_np:[\\pipe\\spoolss]"),
    pointer_default(unique)
]

interface winspool {

// [MS-RPRN] methods
DWORD
RpcEnumPrinters(
    [in] DWORD Flags,
    [in, string, unique] STRING_HANDLE Name,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
        pPrinterEnum,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded,
    [out] DWORD* pcReturned
);

DWORD
RpcOpenPrinter(
    [in, string, unique] STRING_HANDLE pPrinterName,
    [out] PRINTER_HANDLE* pHandle,
    [in, string, unique] wchar_t* pDatatype,
    [in] DEVMODE_CONTAINER* pDevModeContainer,
    [in] DWORD AccessRequired
);

DWORD
RpcSetJob(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD JobId,
    [in, unique] JOB_CONTAINER* pJobContainer,
    [in] DWORD Command
);

DWORD
RpcGetJob(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD JobId,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
        pJob,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded
);

```

```

DWORD
RpcEnumJobs(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD FirstJob,
    [in] DWORD NoJobs,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
        pJob,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded,
    [out] DWORD* pcReturned
);

DWORD
RpcAddPrinter(
    [in, string, unique] STRING_HANDLE pName,
    [in] PRINTER_CONTAINER* pPrinterContainer,
    [in] DEVMODE_CONTAINER* pDevModeContainer,
    [in] SECURITY_CONTAINER* pSecurityContainer,
    [out] PRINTER_HANDLE* pHandle
);

DWORD
RpcDeletePrinter(
    [in] PRINTER_HANDLE hPrinter
);

DWORD
RpcSetPrinter(
    [in] PRINTER_HANDLE hPrinter,
    [in] PRINTER_CONTAINER* pPrinterContainer,
    [in] DEVMODE_CONTAINER* pDevModeContainer,
    [in] SECURITY_CONTAINER* pSecurityContainer,
    [in] DWORD Command
);

DWORD
RpcGetPrinter(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
        pPrinter,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded
);

DWORD
RpcAddPrinterDriver(
    [in, string, unique] STRING_HANDLE pName,
    [in] DRIVER_CONTAINER* pDriverContainer
);

DWORD
RpcEnumPrinterDrivers(
    [in, string, unique] STRING_HANDLE pName,
    [in, string, unique] wchar_t* pEnvironment,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*

```

```

        pDrivers,
        [in] DWORD cbBuf,
        [out] DWORD* pcbNeeded,
        [out] DWORD* pcReturned
    );

    DWORD
    RpcGetPrinterDriver(
        [in] PRINTER_HANDLE hPrinter,
        [in, string, unique] wchar_t* pEnvironment,
        [in] DWORD Level,
        [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
            pDriver,
        [in] DWORD cbBuf,
        [out] DWORD* pcbNeeded
    );

    DWORD
    RpcGetPrinterDriverDirectory(
        [in, string, unique] STRING_HANDLE pName,
        [in, string, unique] wchar_t* pEnvironment,
        [in] DWORD Level,
        [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
            pDriverDirectory,
        [in] DWORD cbBuf,
        [out] DWORD* pcbNeeded
    );

    DWORD
    RpcDeletePrinterDriver(
        [in, string, unique] STRING_HANDLE pName,
        [in, string] wchar_t* pEnvironment,
        [in, string] wchar_t* pDriverName
    );

    DWORD
    RpcAddPrintProcessor(
        [in, string, unique] STRING_HANDLE pName,
        [in, string] wchar_t* pEnvironment,
        [in, string] wchar_t* pPathName,
        [in, string] wchar_t* pPrintProcessorName
    );

    DWORD
    RpcEnumPrintProcessors(
        [in, string, unique] STRING_HANDLE pName,
        [in, string, unique] wchar_t* pEnvironment,
        [in] DWORD Level,
        [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
            pPrintProcessorInfo,
        [in] DWORD cbBuf,
        [out] DWORD* pcbNeeded,
        [out] DWORD* pcReturned
    );

    DWORD
    RpcGetPrintProcessorDirectory(
        [in, string, unique] STRING_HANDLE pName,

```

```

[in, string, unique] wchar_t* pEnvironment,
[in] DWORD Level,
[in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
pPrintProcessorDirectory,
[in] DWORD cbBuf,
[out] DWORD* pcbNeeded
);

DWORD
RpcStartDocPrinter(
[in] PRINTER_HANDLE hPrinter,
[in] DOC_INFO_CONTAINER* pDocInfoContainer,
[out] DWORD* pJobId
);

DWORD
RpcStartPagePrinter(
[in] PRINTER_HANDLE hPrinter
);

DWORD
RpcWritePrinter(
[in] PRINTER_HANDLE hPrinter,
[in, size_is(cbBuf)] BYTE* pBuf,
[in] DWORD cbBuf,
[out] DWORD* pcWritten
);

DWORD
RpcEndPagePrinter(
[in] PRINTER_HANDLE hPrinter
);

DWORD
RpcAbortPrinter(
[in] PRINTER_HANDLE hPrinter
);

DWORD
RpcReadPrinter(
[in] PRINTER_HANDLE hPrinter,
[out, size_is(cbBuf)] BYTE* pBuf,
[in] DWORD cbBuf,
[out] DWORD* pcNoBytesRead
);

DWORD
RpcEndDocPrinter(
[in] PRINTER_HANDLE hPrinter
);

DWORD
RpcAddJob(
[in] PRINTER_HANDLE hPrinter,
[in] DWORD Level,
[in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
pAddJob,
[in] DWORD cbBuf,

```

```

        [out] DWORD* pcbNeeded
    );

    DWORD
    RpcScheduleJob(
        [in] PRINTER_HANDLE hPrinter,
        [in] DWORD JobId
    );

    DWORD
    RpcGetPrinterData(
        [in] PRINTER_HANDLE hPrinter,
        [in, string] wchar_t* pValueName,
        [out] DWORD* pType,
        [out, size_is(nSize)] BYTE* pData,
        [in] DWORD nSize,
        [out] DWORD* pcbNeeded
    );

    DWORD
    RpcSetPrinterData(
        [in] PRINTER_HANDLE hPrinter,
        [in, string] wchar_t* pValueName,
        [in] DWORD Type,
        [in, size_is(cbData)] BYTE* pData,
        [in] DWORD cbData
    );

    DWORD
    RpcWaitForPrinterChange(
        [in] PRINTER_HANDLE hPrinter,
        [in] DWORD Flags,
        [out] DWORD* pFlags
    );

    DWORD
    RpcClosePrinter(
        [in, out] PRINTER_HANDLE*phPrinter
    );

    DWORD
    RpcAddForm(
        [in] PRINTER_HANDLE hPrinter,
        [in] FORM_CONTAINER* pFormInfoContainer
    );

    DWORD
    RpcDeleteForm(
        [in] PRINTER_HANDLE hPrinter,
        [in, string] wchar_t* pFormName
    );

    DWORD
    RpcGetForm(
        [in] PRINTER_HANDLE hPrinter,
        [in, string] wchar_t* pFormName,
        [in] DWORD Level,
        [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*

```

```

        pForm,
        [in] DWORD cbBuf,
        [out] DWORD* pcbNeeded
    );

    DWORD
    RpcSetForm(
        [in] PRINTER_HANDLE hPrinter,
        [in, string] wchar_t* pFormName,
        [in] FORM_CONTAINER* pFormInfoContainer
    );

    DWORD
    RpcEnumForms(
        [in] PRINTER_HANDLE hPrinter,
        [in] DWORD Level,
        [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
            pForm,
        [in] DWORD cbBuf,
        [out] DWORD* pcbNeeded,
        [out] DWORD* pcReturned
    );

    DWORD
    RpcEnumPorts(
        [in, string, unique] STRING_HANDLE pName,
        [in] DWORD Level,
        [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
            pPort,
        [in] DWORD cbBuf,
        [out] DWORD* pcbNeeded,
        [out] DWORD* pcReturned
    );

    DWORD
    RpcEnumMonitors(
        [in, string, unique] STRING_HANDLE pName,
        [in] DWORD Level,
        [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
            pMonitor,
        [in] DWORD cbBuf,
        [out] DWORD* pcbNeeded,
        [out] DWORD* pcReturned
    );

    void
    Opnum37NotUsedOnWire();

    void
    Opnum38NotUsedOnWire();

    DWORD
    RpcDeletePort(
        [in, string, unique] STRING_HANDLE pName,
        [in] ULONG_PTR hWnd,
        [in, string] wchar_t* pPortName
    );

```

```

DWORD
RpcCreatePrinterIC(
    [in] PRINTER_HANDLE hPrinter,
    [out] GDI_HANDLE* pHandle,
    [in] DEVMODE_CONTAINER* pDevModeContainer
);

DWORD
RpcPlayGdiScriptOnPrinterIC(
    [in] GDI_HANDLE hPrinterIC,
    [in, size_is(cIn)] BYTE* pIn,
    [in] DWORD cIn,
    [out, size_is(cOut)] BYTE* pOut,
    [in] DWORD cOut,
    [in] DWORD ul
);

DWORD
RpcDeletePrinterIC(
    [in, out] GDI_HANDLE* phPrinterIC
);

void
Opnum43NotUsedOnWire();

void
Opnum44NotUsedOnWire();

void
Opnum45NotUsedOnWire();

DWORD
RpcAddMonitor(
    [in, string, unique] STRING_HANDLE Name,
    [in] MONITOR_CONTAINER* pMonitorContainer
);

DWORD
RpcDeleteMonitor(
    [in, string, unique] STRING_HANDLE Name,
    [in, string, unique] wchar_t* pEnvironment,
    [in, string] wchar_t* pMonitorName
);

DWORD
RpcDeletePrintProcessor(
    [in, string, unique] STRING_HANDLE Name,
    [in, string, unique] wchar_t* pEnvironment,
    [in, string] wchar_t* pPrintProcessorName
);

void
Opnum49NotUsedOnWire();

void
Opnum50NotUsedOnWire();

DWORD

```

```

RpcEnumPrintProcessorDatatypes(
    [in, string, unique] STRING_HANDLE pName,
    [in, string, unique] wchar_t* pPrintProcessorName,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
        pDataatypes,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded,
    [out] DWORD* pcReturned
);

DWORD
RpcResetPrinter(
    [in] PRINTER_HANDLE hPrinter,
    [in, string, unique] wchar_t* pDataatype,
    [in] DEVMODE_CONTAINER* pDevModeContainer
);

DWORD
RpcGetPrinterDriver2(
    [in] PRINTER_HANDLE hPrinter,
    [in, string, unique] wchar_t* pEnvironment,
    [in] DWORD Level,
    [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
        pDriver,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded,
    [in] DWORD dwClientMajorVersion,
    [in] DWORD dwClientMinorVersion,
    [out] DWORD* pdwServerMaxVersion,
    [out] DWORD* pdwServerMinVersion
);

void
Opnum54NotUsedOnWire();

void
Opnum55NotUsedOnWire();

DWORD
RpcFindClosePrinterChangeNotification(
    [in] PRINTER_HANDLE hPrinter
);

void
Opnum57NotUsedOnWire();

DWORD
RpcReplyOpenPrinter(
    [in, string] STRING_HANDLE pMachine,
    [out] PRINTER_HANDLE*phPrinterNotify,
    [in] DWORD dwPrinterRemote,
    [in] DWORD dwType,
    [in, range(0, 512)] DWORD cbBuffer,
    [in, unique, size_is(cbBuffer), disable_consistency_check] BYTE*
        pBuffer
);

```

```

DWORD
RpcRouterReplyPrinter(
    [in] PRINTER_HANDLE hNotify,
    [in] DWORD fdwFlags,
    [in, range(0, 512)] DWORD cbBuffer,
    [in, unique, size_is(cbBuffer), disable_consistency_check] BYTE*
        pBuffer
);

DWORD
RpcReplyClosePrinter(
    [in, out] PRINTER_HANDLE*phNotify
);

DWORD
RpcAddPortEx(
    [in, string, unique] STRING_HANDLE pName,
    [in] PORT_CONTAINER* pPortContainer,
    [in] PORT_VAR_CONTAINER* pPortVarContainer,
    [in, string] wchar_t* pMonitorName
);

DWORD
RpcRemoteFindFirstPrinterChangeNotification(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD fdwFlags,
    [in] DWORD fdwOptions,
    [in, string, unique] wchar_t* pszLocalMachine,
    [in, range(0, 512)] DWORD cbBuffer,
    [in, out, unique, size_is(cbBuffer), disable_consistency_check]
        BYTE* pBuffer
);

void
Opnum63NotUsedOnWire();

void
Opnum64NotUsedOnWire();

DWORD
RpcRemoteFindFirstPrinterChangeNotificationEx(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD fdwFlags,
    [in] DWORD fdwOptions,
    [in, string, unique] wchar_t* pszLocalMachine,
    [in] DWORD dwPrinterLocal,
    [in, unique] RPC_V2_NOTIFY_OPTIONS* pOptions
);

DWORD
RpcRouterReplyPrinterEx(
    [in] PRINTER_HANDLE hNotify,
    [in] DWORD dwColor,
    [in] DWORD fdwFlags,
    [out] DWORD* pdwResult,
    [in] DWORD dwReplyType,
    [in, switch_is(dwReplyType)] RPC_V2_UREPLY_PRINTER Reply
);

```

```

DWORD
RpcRouterRefreshPrinterChangeNotification(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD dwColor,
    [in, unique] RPC_V2_NOTIFY_OPTIONS* pOptions,
    [out] RPC_V2_NOTIFY_INFO** ppInfo
);

void
Opnum68NotUsedOnWire();

DWORD
RpcOpenPrinterEx(
    [in, string, unique] STRING_HANDLE pPrinterName,
    [out] PRINTER_HANDLE* pHandle,
    [in, string, unique] wchar_t* pDataType,
    [in] DEVMODE_CONTAINER* pDevModeContainer,
    [in] DWORD AccessRequired,
    [in] SPLCLIENT_CONTAINER* pClientInfo
);

DWORD
RpcAddPrinterEx(
    [in, string, unique] STRING_HANDLE pName,
    [in] PRINTER_CONTAINER* pPrinterContainer,
    [in] DEVMODE_CONTAINER* pDevModeContainer,
    [in] SECURITY_CONTAINER* pSecurityContainer,
    [in] SPLCLIENT_CONTAINER* pClientInfo,
    [out] PRINTER_HANDLE* pHandle
);

DWORD
RpcSetPort(
    [in, string, unique] STRING_HANDLE pName,
    [in, string, unique] wchar_t* pPortName,
    [in] PORT_CONTAINER* pPortContainer
);

DWORD
RpcEnumPrinterData(
    [in] PRINTER_HANDLE hPrinter,
    [in] DWORD dwIndex,
    [out, size_is(cbValueName/sizeof(wchar_t))] wchar_t* pValueName,
    [in] DWORD cbValueName,
    [out] DWORD* pcbValueName,
    [out] DWORD* pType,
    [out, size_is(cbData)] BYTE* pData,
    [in] DWORD cbData,
    [out] DWORD* pcbData
);

DWORD
RpcDeletePrinterData(
    [in] PRINTER_HANDLE hPrinter,
    [in, string] wchar_t* pValueName
);

```

```

void
Opnum74NotUsedOnWire();

void
Opnum75NotUsedOnWire();

void
Opnum76NotUsedOnWire();

DWORD
RpcSetPrinterDataEx(
    [in] PRINTER_HANDLE hPrinter,
    [in, string] const wchar_t* pKeyName,
    [in, string] const wchar_t* pValueName,
    [in] DWORD Type,
    [in, size_is(cbData)] BYTE* pData,
    [in] DWORD cbData
);

DWORD
RpcGetPrinterDataEx(
    [in] PRINTER_HANDLE hPrinter,
    [in, string] const wchar_t* pKeyName,
    [in, string] const wchar_t* pValueName,
    [out] DWORD* pType,
    [out, size_is(nSize)] BYTE* pData,
    [in] DWORD nSize,
    [out] DWORD* pcbNeeded
);

DWORD
RpcEnumPrinterDataEx(
    [in] PRINTER_HANDLE hPrinter,
    [in, string] const wchar_t* pKeyName,
    [out, size_is(cbEnumValues)] BYTE* pEnumValues,
    [in] DWORD cbEnumValues,
    [out] DWORD* pcbEnumValues,
    [out] DWORD* pnEnumValues
);

DWORD
RpcEnumPrinterKey(
    [in] PRINTER_HANDLE hPrinter,
    [in, string] const wchar_t* pKeyName,
    [out, size_is(cbSubkey/sizeof(wchar_t))] wchar_t* pSubkey,
    [in] DWORD cbSubkey,
    [out] DWORD* pcbSubkey
);

DWORD
RpcDeletePrinterDataEx(
    [in] PRINTER_HANDLE hPrinter,
    [in, string] const wchar_t* pKeyName,
    [in, string] const wchar_t* pValueName
);

DWORD
RpcDeletePrinterKey(

```

```

        [in] PRINTER_HANDLE hPrinter,
        [in, string] const wchar_t* pKeyName
    );

void
Opnum83NotUsedOnWire();

DWORD
RpcDeletePrinterDriverEx(
    [in, string, unique] STRING_HANDLE pName,
    [in, string] wchar_t* pEnvironment,
    [in, string] wchar_t* pDriverName,
    [in] DWORD dwDeleteFlag,
    [in] DWORD dwVersionNum
);

DWORD
RpcAddPerMachineConnection(
    [in, string, unique] STRING_HANDLE pServer,
    [in, string] const wchar_t* pPrinterName,
    [in, string] const wchar_t* pPrintServer,
    [in, string] const wchar_t* pProvider
);

DWORD
RpcDeletePerMachineConnection(
    [in, string, unique] STRING_HANDLE pServer,
    [in, string] const wchar_t* pPrinterName
);

DWORD
RpcEnumPerMachineConnections(
    [in, string, unique] STRING_HANDLE pServer,
    [in, out, unique, size_is(cbBuf), disable_consistency_check] BYTE*
        pPrinterEnum,
    [in] DWORD cbBuf,
    [out] DWORD* pcbNeeded,
    [out] DWORD* pcReturned
);

DWORD
RpcXcvData(
    [in] PRINTER_HANDLE hXcv,
    [in, string] const wchar_t* pszDataName,
    [in, size_is(cbInputData)] BYTE* pInputData,
    [in] DWORD cbInputData,
    [out, size_is(cbOutputData)] BYTE* pOutputData,
    [in] DWORD cbOutputData,
    [out] DWORD* pcbOutputNeeded,
    [in, out] DWORD* pdwStatus
);

DWORD
RpcAddPrinterDriverEx(
    [in, string, unique] STRING_HANDLE pName,
    [in] DRIVER_CONTAINER* pDriverContainer,
    [in] DWORD dwFileCopyFlags
);

```

```

void
Opnum90NotUsedOnWire();

void
Opnum91NotUsedOnWire();

void
Opnum92NotUsedOnWire();

void
Opnum93NotUsedOnWire();

void
Opnum94NotUsedOnWire();

void
Opnum95NotUsedOnWire();

DWORD
RpcFlushPrinter(
    [in] PRINTER_HANDLE hPrinter,
    [in, size_is(cbBuf)] BYTE* pBuf,
    [in] DWORD cbBuf,
    [out] DWORD* pcWritten,
    [in] DWORD cSleep
);

DWORD RpcSendRecvBidiData(
    [in] PRINTER_HANDLE hPrinter,
    [in, string, unique] const wchar_t* pAction,
    [in] RPC_BIDI_REQUEST_CONTAINER* pReqData,
    [out] RPC_BIDI_RESPONSE_CONTAINER** ppRespData);

void
Opnum98NotUsedOnWire();

void
Opnum99NotUsedOnWire();

void
Opnum100NotUsedOnWire();

void
Opnum101NotUsedOnWire();

void
Opnum102NotUsedOnWire();

void
Opnum103NotUsedOnWire();

void
Opnum104NotUsedOnWire();

void
Opnum105NotUsedOnWire();

```

```
void  
Opnum106NotUsedOnWire();  
  
void  
Opnum107NotUsedOnWire();  
  
void  
Opnum108NotUsedOnWire();  
  
void  
Opnum109NotUsedOnWire();  
}
```

7 Appendix B: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows 95
- Windows 95
- Windows 98
- Windows Server 2003
- Windows Me
- Windows NT 3.51
- Windows NT 4.0
- Windows 2000
- Windows XP
- Windows Vista
- Windows Server 2008

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

[<1> Section 1.3.2:](#) Windows uses various spool file formats, such as Enhanced Metafile (EMF) Spool Format or RAW format.

In Windows Vista and Windows Server 2008, the **XML Paper Specification** format may be used also. For more information on these formats, see [\[MSDN-SPOOL\]](#), [\[MS-EMFSPOOL\]](#) and [\[MSDN-XMLP\]](#).

[<2> Section 1.6:](#) Active Directory (AD) manages the client configuration. For more information about AD, see [\[MSDN-ADOVRVW\]](#).

[<3> Section 1.8:](#) Windows implementations of this protocol use only the values specified in [\[MS-ERREF\]](#).

[<4> Section 2.1:](#) The callee impersonates the caller when processing a method.

Windows Vista and Windows Server 2008 register the "Security Provider Simple and Protected GSS-API Negotiation Mechanism (SNEGO)" **Security Provider**.

Windows Server 2003 registers the "NT LAN Manager (NTLM)" Security Provider.

Windows 2000, Windows NT 4.0, Windows NT 3.5, Windows NT 3.51, and Windows NT 3.1 do not register a Security Provider.

[<5> Section 2.2.1.1.7:](#) The Windows print server obtains this checksum by calling the **GdiQueryFonts** API method.

<6> [Section 2.2.1.3.1:](#) The Windows operating system uses the following values to indicate printer drivers on different OS versions:

Value	Meaning
0x00000000	Printer driver for Windows Me, Windows 98, and Windows 95.
0x00000001	Printer driver for Windows NT 3.51.
0x00000002	Kernel-mode printer driver for Windows NT 4.0.
0x00000003	User-mode printer driver for Windows Vista, Windows Server 2003, Windows XP, and Windows 2000.

<7> [Section 2.2.1.3.1:](#) For example, "C:\DRIVERS\Pscript.dll".

<8> [Section 2.2.1.3.1:](#) For example, the full path for the printer driver data binary could be specified as "C:\DRIVERS\Qms810.ppd".

<9> [Section 2.2.1.3.1:](#) Windows uses the printer driver configuration dynamic-link library (DLL), such as "C:\DRIVERS\Pscriptui.dll".

<10> [Section 2.2.1.3.1:](#) For example, the full path for the printer driver's help file could be specified as "C:\DRIVERS\PscriptUx.dll".

<11> [Section 2.2.1.3.1:](#) A language monitor is specified for printers capable of bidirectional communication. The name is specific to a printer manufacturer. For example, the name of a language monitor could be specified as "PJM monitor".

<12> [Section 2.2.1.3.1:](#) The following is an example of the list of dependent files:
"Pscript.dll\0Qms810.PPD\0Pscriptui.dll\0Pscriptui.hlp\0Pstest.txt\0\0".

<13> [Section 2.2.1.3.1:](#) For example, "OldName1\0OldName2\0\0".

<14> [Section 2.2.1.3.1:](#) Windows operating systems use a combination of the OS major and minor numbers, the build number, and revision. For example, the printer driver version number of 0x000500020ECE0726 represents:

- OS Major Version: 0x0005
- OS Minor Version: 0x0002
- Build number: 0x0ECE (3790)
- Revision: 0x0726 (1830)

<15> [Section 2.2.1.3.1:](#) For example, "ROOT\FTDISK".

<16> [Section 2.2.1.3.1:](#) For example, "Microsoft".

<17> [Section 2.2.1.3.6:](#) If non-null, the string length must be less than or equal to 1041 characters.

<18> [Section 2.2.1.3.6:](#) If non-null, the string length must be less than or equal to 256 characters.

<19> [Section 2.2.1.3.7:](#) The Windows operating system uses the following values:

Value	Meaning
0x00000004	The operating system is Windows NT 4.0, Windows Me, Windows 98, or Windows 95.
0x00000005	The operating system is Windows Server 2003, Windows XP, or Windows 2000.
0x00000006	The operating system is Windows Vista.

<20> [Section 2.2.1.3.7:](#) The Windows operating system uses the following values:

Value	Meaning
0x00000000	The operating system is Windows Vista, Windows Server 2008, Windows 2000, Windows NT 4.0, or Windows 95.
0x00000001	The operating system is Windows XP.
0x00000002	The operating system is Windows Server 2003 or Windows XP Professional x64 Edition.
0x0000000A	The operating system is Windows 98.
0x0000005A	The operating system is Windows Me.

<21> [Section 2.2.1.3.7:](#) Windows uses the following values:

Name/Value	Meaning
PROCESSOR_ARCHITECTURE_INTEL 0x0000	X86 architecture
PROCESSOR_ARCHITECTURE_IA64 0x0006	Itanium architecture
PROCESSOR_ARCHITECTURE_AMD64 0x0009	Amd64 architecture

<22> [Section 2.2.1.5.6:](#) **INF files** are used to define the printer driver configuration.

<23> [Section 2.2.1.5.6:](#) INF files are used to define the printer driver configuration.

<24> [Section 2.2.1.5.6:](#) This value is read from the printer driver INF file.

<25> [Section 2.2.1.5.6:](#) This value is read from the printer driver INF file.

<26> [Section 2.2.1.6.2:](#) Printer Drivers generate an implementation-specific unique identifier. Windows print servers generate a unique GUID identifier.

<27> [Section 2.2.1.9.2:](#) This member contains a descriptive name for the port monitor. For example: "Standard TCP/IP Port", "Fax Monitor Port", or "Local Port"

<28> [Section 2.2.1.10.1:](#) Windows calculates the version by storing the build version in the high-order 16 bits, and the operation system release number in the low-order 16 bits.

Note: For example, 0x0A280005 corresponds to XP build 2600.

<29> [Section 2.2.1.10.1:](#) The debugging build of the print server sets **fFreeBuild** to 0, and the release build of the print server sets **fFreeBuild** to 1.

<30> [Section 2.2.1.10.1:](#) Supported Processor Types

Windows uses the following values:

Value	Meaning
PROCESSOR_INTEL_386 0x00000182	Intel 80386 compatible
PROCESSOR_INTEL_486 0x000001E6	Intel 80486 compatible
PROCESSOR_INTEL_PENTIUM 0x0000024A	Intel Pentium compatible
PROCESSOR_INTEL_IA64 0x00000898	Intel IA64 compatible
PROCESSOR_AMD_X8664 0x000022A0	AMD x64 compatible

<31> [Section 2.2.1.10.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<32> [Section 2.2.1.10.1:](#) Windows uses the following values:

Value	Meaning
PROCESSOR_ARCHITECTURE_INTEL 0x0000	X86 architecture
PROCESSOR_ARCHITECTURE_IA64 0x0006	Itanium architecture
PROCESSOR_ARCHITECTURE_AMD64 0x0009	AMD64 architecture

<33> [Section 2.2.1.10.1:](#) Windows uses the value of 1 for PROCESSOR_ARCHITECTURE_IA64 and PROCESSOR_ARCHITECTURE_AMD64.

For PROCESSOR_ARCHITECTURE_INTEL, Windows uses the value defined by the CPU vendor.

<34> [Section 2.2.1.10.2:](#) A Windows operating system will display this printer in its list of network available printers.

<35> [Section 2.2.1.10.2:](#) A Windows operating system will display this printer in its list of network available printers.

<36> [Section 2.2.1.10.2:](#) A Windows operating system will display this printer in its list of network available printers.

<37> [Section 2.2.1.10.8:](#) The string representation of a 128-bit GUID is in the form of a **GUIDString**, defined in [\[MS-GLOS\]](#).

[<38> Section 2.2.1.12.1:](#) The bidirectional communications schema is a hierarchy of printer attributes, some of which are properties, with the rest being values (or value entries). Bidirectional communications interfaces are implemented by printer-specific components. A detailed description of printer drivers and the bidirectional communications schema can be found in the Windows Device Driver Kit.

[<39> Section 2.2.1.12.2:](#) Windows returns a nonzero error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<40> Section 2.2.1.12.2:](#) The bidirectional communications schema is a hierarchy of printer attributes, some of which are properties, with the rest being values (or value entries). Bidirectional communications interfaces are implemented by printer-specific components. A detailed description of printer drivers and the bidirectional communications schema can be found in the Windows Device Driver Kit.

[<41> Section 2.2.1.12.3:](#) The Windows system will not localize the bidirectional data.

[<42> Section 2.2.2.1:](#) Versions of initialization data specifications correspond to versions of Windows operating systems as follows:

Value	Meaning
0x0320	Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51.
0x0400	Windows 95, Windows 98, and Windows Me.
0x0401	Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008.

[<43> Section 2.2.2.1:](#) Versions of printer drivers correspond to versions of Windows operating systems as follows:

Value	Meaning
0x0301	Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51 user-mode printer drivers, and Windows NT 4.0 kernel-mode printer drivers.
0x0500	Windows 2000, Windows XP, and Windows Server 2003 user-mode printer drivers.
0x0600	Windows Vista and Windows Server 2008 printer drivers.

[<44> Section 2.2.2.1:](#) The value of this member is set by printer manufacturers, depending on the needs of the printer driver.

[<45> Section 2.2.2.1.1:](#) **PSCRIPT** is the Windows PostScript core printer driver. It stores the private data that it needs in the **Driver Extra Data** area by using the following structures.

The **dmDriverExtraData** area contains one of the **PSDRVEXTRA** structures, immediately followed by zero or one **JTEXP** structures (see section [2.2.2.1.4](#)), which is then followed by zero or more **OEM_DMEXTRA** structures (see [2.2.2.1.3](#)).

These structures are not part of the protocol defined in this document, and they are subject to change without notice. Implementations of compatible drivers must check the structure version and discard any **Driver Extra Data** they do not handle.

PSDRVEXTRA351 – defined by PSCRIPT Driver released with Windows NT 3.51:

This structure is used for **PSCRIPT Driver Extra Data** if the **dmDriverVersion** of the [DEVMODE](#) structure is 0x0350.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
dwSignature																															
dwFlags																															
wchEPSFile																															
...																															
...																															
...																															
...																															
...																															
...																															
(wchEPSFile cont'd for 12 rows)																															
caSize																caFlags															
caIlluminantIndex																caRedGamma															
caGreenGamma																caBlueGamma															
caReferenceBlack																caReferenceWhite															
caContrast																caBrightness															
caColorfulness																caRedGreenTint															

PSDRVEXTRA400 – defined by PSCRIPT Driver released with Windows NT 4.0:

This structure is used for PSCRIPT Driver Extra Data if the **dmDriverVersion** of the [DEVMODE](#) structure is 0x0400.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
dwSignature																															
dwFlags																															
wchEPSFile																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															
(wchEPSFile cont'd for 12 rows)																															
caSize																caFlags															
caIlluminantIndex																caRedGamma															
caGreenGamma																caBlueGamma															
caReferenceBlack																caReferenceWhite															
caContrast																caBrightness															
caColorfulness																caRedGreenTint															
wChecksum																wOptions															
aubOptions																															

...
...
...
...
...
...
...
...
(aubOptions cont'd for 8 rows)

wChecksum: The value of this field is a checksum of the aubOptions array.

wOptions: The value of this field is the number of entries in the aubOptions array that are initialized.

aubOptions: This field is an array of 64 bytes in length and contains user interface selections. Unused fields should be initialized to zero. The meaning of the entries in this array differs for each supported printer model. Upon receipt, the checksum of this array is computed and compared to wChecksum. The array is used only if the checksums match.

PSDRVEXTRA500 – defined by PSCRIPT Driver released with Windows 2000 and Windows XP:

This structure is used for PSCRIPT Driver Extra Data if the **dmDriverVersion** of the [DEVMODE](#) structure is 0x0501.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
dwSignature																															
dwFlags																															
wchEPSFile																															
...																															
...																															
...																															

...	
...	
...	
...	
(wchEPSFile cont'd for 12 rows)	
caSize	caFlags
caIlluminantIndex	caRedGamma
caGreenGamma	caBlueGamma
caReferenceBlack	caReferenceWhite
caContrast	caBrightness
caColorfulness	caRedGreenTint
wReserved1	wSize
fxScrFreq	
fxScrAngle	
iDialect	
iTtDLFmt	
bReversePrint	
iLayout	
iPSLevel	
dwReserved2	
wOEMExtra	wVer

dwX	
dwY	
dwWidthOffset	
dwHeightOffset	
wFeedDirection	wCutSheet
dwReserved3	
...	
...	
...	
dwChecksum32	
dwOptions	
aOptions[0]	
...	
(aOptions array cont'd for 126 elements)	

wReserved1: The value of this field should be set to zero and it must be ignored upon receipt.

wSize: This field is the same as wCoreFullSize in PSDRVEXTRA.

PSDRVEXTRA – defined by PSCRIPT Driver released with Windows Vista **and** Windows Server 2008:

This structure is used for PSCRIPT Driver Extra Data if the **dmDriverVersion** of the [DEVMODE](#) structure is 0x0600.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
dwSignature																															

dwFlags	
wchEPSFile	
...	
...	
...	
...	
...	
...	
...	
(wchEPSFile cont'd for 12 rows)	
caSize	caFlags
caIlluminantIndex	caRedGamma
caGreenGamma	caBlueGamma
caReferenceBlack	caReferenceWhite
caContrast	caBrightness
caColorfulness	caRedGreenTint
wCoreJTExpSize	wCoreFullSize
fxScrFreq	
fxScrAngle	
iDialect	
iTDLFmt	

bReversePrint	
iLayout	
iPSLevel	
dwReserved2	
wOEMExtra	wVer
dwX	
dwY	
dwWidthOffset	
dwHeightOffset	
wFeedDirection	wCutSheet
dwReserved3	
...	
...	
...	
dwChecksum32	
dwOptions	
aOptions[0]	
...	
(aOptions array cont'd for 126 elements)	
dwNupDirection	
dwNupBorderFlags	

dwBookletFlags
dwPadding

wCoreJTEpSize: The value of this field specifies the size of the **JTEXP** structure if one is present, following directly after this structure.

wCoreFullSize: The value of this field specifies the size of the **PSDRVEXTRA** structure plus the value of **wCoreJTEpSize**.

dwNupDirection: This field is used only if N-Up printing is selected, and the value of this field must be one of the following:

Value	Meaning
0x00000001	Print N-Up pages left-to-right, top-to-bottom.
0x00000002	Print N-Up pages top-to-bottom, left-to-right.
0x00000004	Print N-Up pages right-to-left, top-to-bottom.
0x00000008	Print N-Up pages top-to-bottom, right-to-left.

dwNupBorderFlags: This field is used only if N-Up printing is selected, and the value of this field must be one of the following:

Value	Meaning
0x00000000	Print borders around N-Up pages.
0x00000001	Do not print borders around N-Up pages.

dwBookletFlags: This field is used only if booklet printing is selected, and the value of this field must be one of the following:

Value	Meaning
0x00000000	Print booklet so that pages flip to the left (western style).
0x00000001	Print booklet so that pages flip to the right.

dwPadding: The value of this field should be set to zero and it must be ignored upon receipt.

Description of Common Members for PSDRVEXTRA Structures:

dwSignature: The value of this field must be 0x56495250.

dwFlags: The value of this field must be the result of the bitwise OR of zero or more of the following values:

Value	Meaning
0x00000002	Send PostScript driver error handler code.

Value	Meaning
0x00000004	Print mirror image.
0x00000010	Print negative image of page.
0x00000040	Compress bitmaps.
0x00000200	If DEVMODE member dmOrientation is set to 2 (LANDSCAPE), rotate page by an additional 180 degrees.
0x00002000	If driver is initialized by GDI, inform GDI that metafile spooling is requested.

wchEPSFile: This field is not used.

caSize: The value of this field must be 24.

caFlags: This field is not used.

caIlluminantIndex: This field is not used.

caRedGamma: This field is not used.

caGreenGamma: This field is not used.

caBlueGamma: This field is not used.

caReferenceBlack: This field is not used.

caReferenceWhite: This field is not used.

caContrast: This field is not used.

caBrightness: This field is not used.

caColorfulness: This field is not used.

caRedGreenTint: This field is not used.

fxScrFreq: This field is not used.

fxScrAngle: This field is not used.

iDialect: The value of this field must be one of the following:

Value	Meaning
0x00000000	Optimize generated PostScript for speed.
0x00000001	Optimize generated PostScript for portability.
0x00000002	Optimize generated PostScript for EPS use.
0x00000003	Optimize generated PostScript for archival.

iTTDLFmt: The value of this field must be one of the following:

Value	Meaning
0x00000000	Download fonts in default format, pick the best suited format for font.
0x00000001	Download fonts as Type 1 outlines.
0x00000002	Download fonts as Type 3 bitmaps.
0x00000003	Download fonts as Type 42 fonts.
0x00000004	Same as 0x00000000.
0x00000005	Same as 0x00000000.

bReversePrint: If the value of this field is nonzero, print pages in reverse order; otherwise, print pages in normal order.

iLayout: The value of this field must be one of the following:

Value	Meaning
0x00000000	N-Up printing disabled.
0x00000001	Print 2-Up.
0x00000002	Print 4-Up.
0x00000003	Print 6-Up.
0x00000004	Print 9-Up.
0x00000005	Print 16-Up.
0x00000006	Print job as booklet.

iPSLevel: The value of this field must be one of the following:

Value	Meaning
0x00000001	Use only PostScript level 1 features.
0x00000002	Use only PostScript level 1 and level 2 features.
0x00000003	Use all PostScript features available for level 1, level 2, and level 3.

dwReserved2: The value of this field should be set to zero, and it must be ignored upon receipt.

wOEMExtra: The value of this field must be the total size of the private Driver Extra Data that is used by the vendor-supplied driver **plug-in**. This data immediately follows this structure.

wVer: The value of this field must be 0x0010.

dwX: The value of this field specifies the width, in 1/1000th millimeter units, of the custom paper size. Used only if the **dmPaperSize** member of [DEVMODE](#) is set to 0x7FFF.

dwY: The value of this field specifies the height, in 1/1000th millimeter units, of the custom paper size. Used only if the **dmPaperSize** member of [DEVMODE](#) is set to 0x7FFF.

dwWidthOffset: The value of this field specifies the left unprintable margin, in 1/1000th of a millimeter, of the custom paper size. Used only if the **dmPaperSize** member of [DEVMODE](#) is set to 0x7FFF.

dwHeightOffset: The value of this field specifies the top unprintable margin, in 1/1000th of a millimeter, of the custom paper size. Used only if the **dmPaperSize** member of [DEVMODE](#) is set to 0x7FFF.

wFeedDirection: The value of this field must be one of the following:

Value	Meaning
0x0000	The paper is physically fed into the print mechanism with its long edge first.
0x0001	The paper is physically fed into the print mechanism with its short edge first.
0x0002	The paper is physically fed into the print mechanism with its long edge first, but upside down.
0x0003	The paper is physically fed into the print mechanism with its long edge first, but upside down.

wCutSheet: The value of this field is zero for roll-fed custom paper sizes and nonzero for cut sheet custom paper. This field is used only if the **dmPaperSize** member of [DEVMODE](#) is set to 0x7FFF.

dwReserved3: The value of this field should be set to zero, and it must be ignored upon receipt.

dwChecksum32: The value of this field is the checksum of the **aOptions** array.

dwOptions: The value of this field specifies the number of entries of the **aOptions** array that are initialized.

aOptions: This field is an array that is 512 bytes long and contains user interface selections. Unused fields should be initialized to zero. The meaning of the entries differs for each supported printer model. Upon receipt, the checksum of this array is computed and compared to

dwChecksum32. The array is used only if the checksums match.

[<46> Section 2.2.2.1.2:](#) **UNIDRV** is the generic Windows core printer driver for all printers that do not use PostScript. It stores its private data in the **Driver Extra Data** area using the following structures.

The **dmDriverExtraData** area contains one of the **UNIDRVEXTRA** structures, immediately followed by zero or one **JTEXP** (see section [2.2.2.1.4](#)), followed by zero or more **OEM_DMEXTRA** structures (see [2.2.2.1.3](#)).

These structures are not part of the protocol defined in this document, and they are subject to change without notice. Implementations of compatible drivers must check the structure version and discard any **Driver Extra Data** they do not handle.

UNIDRVEXTRA3_4 – defined by UNIDRV Driver released with Windows NT 4.0 and Windows NT 3.5:

This structure is used for **UNIDRV Driver Extra Data** if the **dmDriverVersion** of the [DEVMODE](#) structure is 0x0301.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
wReserved[0]																...															
...																(repeats for total of 56 reserved words)															

wReserved: This field is no longer used by **UNIDRV**.

UNIDRVEXTRA500 – defined by **UNIDRV** driver released with Windows XP and Windows 2000:

This structure is used for **UNIDRV Driver Extra Data** if the **dmDriverVersion** of the [DEVMODE](#) structure is 0x0500.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
dwSignature																															
wVer																sPadding															
wSize																wOEMExtra															
dwChecksum32																															
dwFlags																															
bReversePrint																															
iLayout																															
iQuality																															
wReserved																...															
...																...															
...																...															
dwOptions																															
aOptions[0]																															

...
(repeats for a total of 128 aOptions array elements)
aOptions[127]

sPadding: The value of this field should be set to zero, and it must be ignored upon receipt.

wSize: This field is the same as wCoreFullSize in UNIDRVEXTRA.

UNIDRVEXTRA – defined by UNIDRV Driver released with Windows Server 2008 and Windows Vista:

This structure is used for UNIDRV Driver Extra Data if the **dmDriverVersion** of the [DEVMODE](#) structure is 0x0600.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
dwSignature																															
wVer																wCoreJTExpSize															
wCoreFullSize																wOEMExtra															
dwChecksum32																															
dwFlags																															
bReversePrint																															
iLayout																															
iQuality																															
wReserved																...															
...																...															
...																...															
dwOptions																															

aOptions[0]
...
(repeats for a total of 128 aOptions array elements)
aOptions[127]
dwNupDirection
dwNupBorderFlags
dwBookletFlags

wCoreJTEXpSize: The value of this field specifies the size of the JTEXP if one is present, following directly after this structure.

wCoreFullSize: The value of this field specifies the size of the UNIDRVEXTRA structure plus the value of wCoreJTEXpSize.

dwNupDirection: This field is used only if N-Up printing is selected, and it must be one of the following values:

Value	Meaning
0x00000001	Print N-Up pages left-to-right, top-to-bottom.
0x00000002	Print N-Up pages top-to-bottom, left-to-right.
0x00000004	Print N-Up pages right-to-left, top-to-bottom.
0x00000008	Print N-Up pages top-to-bottom, right-to-left.

dwNupBorderFlags: This field is used only if N-Up Printing is selected, and it must be one of the following values:

Value	Meaning
0x00000000	Print borders around N-Up pages.
0x00000001	Do not print borders around N-Up pages.

dwBookletFlags: This field is used only if booklet printing is selected, and it must be one of the following values:

Value	Meaning
0x00000000	Print booklet so that pages flip to the left (western style).
0x00000001	Print booklet so that pages flip to the right.

Description of Common Members for UNIDRVEXTRA structures:

dwSignature: The value of this field must be 0x44494E55.

dwFlags: The value of this field must be the result of the bitwise OR of zero or more of the following values:

Value	Meaning
0x00000002	Print text as graphics (do not use fonts).
0x00000010	Do not use Enhanced Metafile (EMF) spooling.
0x00000080	Use Custom Quality halftoning.

bReversePrint: If the value of this field is nonzero, print pages in reverse order; otherwise, print pages in normal order.

iLayout: The value of this field must be one of the following values

Value	Meaning
0x00000000	N-Up printing disabled.
0x00000001	Print 2-Up.
0x00000002	Print 4-Up.
0x00000003	Print 6-Up.
0x00000004	Print 9-Up.
0x00000005	Print 16-Up.
0x00000006	Print job as booklet.

iQuality: The value of this field must be one of the following values:

Value	Meaning
0x00000000	Best Quality
0x00000001	Medium Quality
0x00000002	Draft Quality

wReserved: The value of this field should be set to zero, and it must be ignored upon receipt.

wOEMExtra: The value of this field specifies the total size of private **Driver Extra Data** that is used by vendor-supplied driver plug-in. This data immediately follows this structure.

wVer: The value of this field must be 0x0022.

dwChecksum32: The value of this field must be the checksum of the **aOptions** array.

dwOptions: The value of this field must be the number of entries in the **aOptions** array that are initialized.

aOptions: This field is an array that is 512 bytes in length and contains user interface selections. Unused fields should be initialized to zero. The meaning of the entries differs for each supported printer model. Upon receipt, the checksum of this array is computed and compared to **dwChecksum32**. The array is used only if the checksums match.

[<47> Section 2.2.2.1.3:](#) This structure is the **Driver Extra Data** defined by vendor-supplied driver plug-in modules.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
dwSize																															
dwSignature																															
dwVersion																															
Vendor-defined Data (variable)																															

dwSize: Must be the total size of the vendor-defined **Extra Data**.

dwSignature: Must be a vendor-defined unique number.

dwVersion: Must be the version of the vendor-supplied plug-in.

Vendor-defined Data (variable): A variable-length field that holds vendor-defined data.

[<48> Section 2.2.2.1.4:](#) This structure is the **Driver Extra Data** defined by Microsoft to hold print ticket selection information.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
dwSize																															
dwSignature																															
dwVersion																															
wJTHdrSize																wCoreMFOSize															
ModelName (variable)																															
FeatureOptionPairs (variable)																															

dwSize: Must be the total size of **Extra Data**.

dwSignature: Must be 0x534D544A.

dwVersion: Must be zero.

wJTHdrSize: Must be set to 16.

wCoreMFOSize: Must be the combined size of **ModelName** (variable) and **FeatureOptionPairs** (variable).

ModelName (variable): Must be a zero-terminated **UTF-16LE** encoded string specifying the name of the printer model.

FeatureOptionPairs (variable): Must be a concatenation of an even number of zero-terminated ASCII strings, terminated by an additional zero character. Each pair of two consecutive strings specifies a print schema feature and the currently selected option.

[<49> Section 2.2.2.5.7:](#) Windows uses INF files for installation configuration data. For more information, see [\[MSDN-UIINF\]](#) for more details.

[<50> Section 2.2.3.2:](#) Windows does not use this value.

[<51> Section 2.2.3.2:](#) A Windows operating system will display this printer in its list of network-available printers.

[<52> Section 2.2.3.2:](#) A Windows operating system will display this printer in its list of network-available printers.

[<53> Section 2.2.3.2:](#) A Windows operating system will display this printer in its list of network-available printers.

[<54> Section 2.2.3.2:](#) Windows does not use this value.

[<55> Section 2.2.3.2:](#) Windows does not use this value.

[<56> Section 2.2.3.2:](#) Windows does not use this value.

[<57> Section 2.2.3.2:](#) Windows does not use this value.

[<58> Section 2.2.3.2:](#) A Windows operating system will display this printer in its list of network-available printers.

[<59> Section 2.2.3.8:](#) The Windows operating system uses the following key names with their associated registry type values (see section [2.2.3.9](#) for details) to store printer configuration data.

Server Handle Key Name/Registry Type Value	Meaning
"DefaultSpoolDirectory" REG_SZ	Default spooler directory configuration data.
"PortThreadPriorityDefault" REG_DWORD	Port's thread default priority.
"PortThreadPriority" REG_DWORD	Port's thread priority.
"SchedulerThreadPriorityDefault" REG_DWORD	Scheduler's thread default priority.

Server Handle Key Name/Registry Type Value	Meaning
"SchedulerThreadPriority" REG_DWORD	Scheduler's thread priority.
"BeepEnabled" REG_DWORD	Beep enabled.
"NetPopup" REG_DWORD	Configuration data for sending job notifications to the user.
"RetryPopup" REG_DWORD	Retry pop-up windows for all jobs.
"NetPopupToComputer" REG_DWORD	Sending job notifications to the client or the user.
"EventLog" REG_DWORD	Configuration data for the event log.
"MajorVersion" REG_DWORD	Major version.
"MinorVersion" REG_DWORD	Minor version.
"Architecture" REG_SZ	Processor architecture, 32-bit or 64-bit.
"OSVersion" REG_BINARY	Operating system version number.
"OSVersionEx" REG_BINARY	Operating system version number.
"DsPresent" REG_DWORD	0x0001 if the print server is joined to a domain with directory services (DS), zero if not.
"DsPresentForUser" REG_DWORD	0x0001 if the user is logged on to a domain with directory services, zero if not.
"RemoteFax" REG_DWORD	0x0001 if the fax service supports remote clients, zero otherwise.
"RestartJobOnPoolError" REG_DWORD	Minimum time, in seconds, when a job is restarted on another port after an error occurs.
"RestartJobOnPoolEnabled" REG_DWORD	Nonzero value indicates that SPLREG_RESTART_JOB_ON_POOL_ERROR is enabled.
"DNSMachineName" REG_SZ	Domain name service computer name.

[<60> Section 2.2.4.1:](#) The Windows implementation restricts the total length of **SERVER_NAME** to 259 characters, including the two leading backslash characters and trailing backslash character.

IPv6 names are supported only on Windows Server 2008 and Windows Vista.

[<61> Section 2.2.4.2:](#) The Windows implementation restricts the total length of **PRINTER_NAME_EX** to 539 characters (259 + 260 + 20), including all backslashes, other separators, and the terminating null character.

[<62> Section 2.2.4.5:](#) The Windows implementation uses the following environment name strings:

"Windows NT x86"

"Windows IA64"

"Windows x64"

"Windows 4.0".

[<63> Section 2.2.4.6:](#) The Windows implementation uses driver name strings up to 260 characters long, including the terminating null character.

[<64> Section 2.2.4.7:](#) The Windows implementation uses path name strings up to 519 characters, including the terminating null character. The following pattern is used for local files:

NAME = <any TEXT except "\">

DIRECTORY = "\" 1#NAME

FILENAME = "\" 1#NAME

PATH = [alpha ":"] #DIRECTORY FILENAME

IPv6 names are supported only on Windows Server 2008 and Windows Vista.

[<65> Section 2.2.4.8:](#) The Windows implementation uses print processor name strings up to 260 characters, including the terminating null character.

[<66> Section 2.2.4.11:](#) The Windows implementation uses form name strings up to 260 characters, including the terminating null character.

[<67> Section 2.2.4.12:](#) The Windows implementation uses monitor name strings up to 260 characters, including the terminating null character.

[<68> Section 2.2.4.13:](#) The Windows operating system uses the following patterns for port names.

PARALLEL_PORT = "LPT" DIGIT ":"

SERIAL_PORT = "COM" DIGIT ":"

FILE_PORT = "FILE:"

USB_PORT = "USB" 1#DIGIT ":"

UNC_PORT = SERVER_NAME DIRECTORY FILENAME

LOCAL_FILE_PORT = PATH

PORT_NAME = (PARALLEL_PORT | SERIAL_PORT | FILE_PORT | USB_PORT | UNC_PORT | LOCAL_FILE_PORT)

where:

SERVER_NAME: See section [2.2.4.1](#) for details.

DIRECTORY: See section [2.2.4.7](#) for details.

PARALLEL_PORT: Is used for devices attached through the parallel port.

SERIAL_PORT: Is used for devices attached through the serial port.

FILE_PORT: Is used to send data to a file.

USB_PORT: Is used for devices attached through a **USB** port.

UNC_PORT: Is used for network printers attached directly through an IP address or a network address.

The Windows operating system supports pooling of ports. When printing to a printer associated with a pool of ports, the first available port is picked by the print server. Port pooling allows representation of multiple identical physical printers as a single logical printer. Pooled port names are represented as a comma-separated list of port names, for example, "LPT1:,LPT2:". Clients connecting to a Windows print server need to be prepared to handle pooled ports correctly; for example, they cannot rely on individual port names enumerated by the [RpcEnumPorts](#) method to match the string pointed to by the **pPortName** member of a **PRINTER_INFO** structure.

[<69> Section 2.2.4.14:](#) The Windows implementation uses print provider name strings up to 260 characters, including the terminating null character.

[<70> Section 2.2.4.15:](#) In Windows, the value can be "EMF", "XPS_PASS", "RAW", or a custom data type defined by a custom print processor. For more information on these formats, see [\[MSDN-SPOOL\]](#), [\[MS-EMFSPool\]](#) and [\[MSDN-XMLP\]](#)

[<71> Section 2.2.4.16:](#) The Windows implementation uses value name strings up to 260 characters, including the terminating null.

[<72> Section 2.2.4.17:](#) The Windows implementation uses key name strings up to 260 characters, including the terminating null character.

[<73> Section 3.1.1:](#) All objects are stored persistently on disk or in the registry. Transient cached copies of persisted objects are maintained while the print server is running to improve system throughput and reduce latency.

[<74> Section 3.1.1:](#) **Persistence of Abstract Data Model in the Registry**

Print system management tools such as **printmig**, **brm**, and other, third-party tools read and write registry values that are used by the Windows print server to persist its abstract data model. Windows print server also persists objects from the abstract data model and other settings in the registry, as described in the following table.

Note that the value of Registry Key entry that is shown in the table below is word-wrapped at the backslash character for easier reading. The actual string value is one contiguous string.

For example: A value in the table such as:

```

HKLM\
SYSTEM\
CurrentControlSet\
Control\
Print

```

represents a registry key path string that would be entered in a string variable as:

```
"HKLM\SYSTEM\CurrentControlSet\Control\Print"
```

Registry key	Value	Type	Description
HKLM\ SYSTEM\ CurrentControlSet\ Control\ Print\ 			
	MaxRPCSize	REG_DWORD	Maximum buffer size accepted by the server in RPC calls. The default used, if this value is not present, is 50MB.
HKLM\ SYSTEM\ CurrentControlSet\ Control\ Print\ Environments\ [Env Name]\ Drivers\ [Version]\ [Driver Name]			Information about the printer driver named [Driver Name] having the version identified by [Version] for the environment named [Env Name].
	Configuration File	REG_SZ	Persisted DRIVER_INFO member pConfigFile .
	Data File	REG_SZ	Persisted DRIVER_INFO member pDataFile .
	Driver	REG_SZ	Persisted DRIVER_INFO member pDriverPath .
	Help File	REG_SZ	Persisted DRIVER_INFO member pHelpFile .
	Monitor	REG_SZ	Persisted DRIVER_INFO member pMonitorName .

Registry key	Value	Type	Description
	Datatype	REG_SZ	Persisted DRIVER_INFO member pDefaultDataType .
	Dependent Files	REG_MULTI_SZ	Persisted DRIVER_INFO member pDependentFiles .
	Previous Names	REG_MULTI_SZ	Persisted DRIVER_INFO member pszPreviousNames .
	Version	REG_DWORD	Persisted DRIVER_INFO member cVersion .
	Manufacturer	REG_SZ	Persisted DRIVER_INFO member pMfgName .
	OEM URL	REG_SZ	Persisted DRIVER_INFO member pOEMUrl .
	HardwareID	REG_SZ	Persisted DRIVER_INFO member pHardwareID .
	Provider	REG_SZ	Persisted DRIVER_INFO member pProvider .
	DriverDate	REG_BINARY	Persisted DRIVER_INFO member ftDriverDate .
	DriverVersion	REG_BINARY	Persisted DRIVER_INFO member dwDriverVersion .
HKLM\ SYSTEM\ CurrentControlSet\ Control\ Print\ Environments\ [Env Name]\ Print Processors\ [Processor Name]			Information about the print processor named [Processor Name] for the environment named [Env Name].
	Driver	REG_SZ	Persisted pPathName parameter used when print processor was added by the RpcAddPrintProcessor method.
HKLM\ SYSTEM\ CurrentControlSet\ Control\ 			Information about the port monitor named [Monitor Name].

Registry key	Value	Type	Description
Print\ Monitors\ [Monitor Name]			
	Driver	REG_SZ	Persisted MONITOR_INFO_2 member pDLLName .
HKLM\ SYSTEM\ CurrentControlSet\ Control\ Print\ Monitors\ LPR Port\ Ports			Information about LPR ports.
HKLM\ SYSTEM\ CurrentControlSet\ Control\ Print\ Monitors\ LPR Port\ Ports\ [Port name]			Information about the LPR port named [Port Name].
	Server Name	REG_SZ	Host address of server exposing LPR port.
	Printer Name	REG_SZ	Queue name on LPR server.
HKLM\ SYSTEM\ CurrentControlSet\ Control\ Print\ Monitors\ Standard TCP/IP Port\ Ports			
HKLM\ SOFTWARE\ DigitalEquipmentCorporation\ Network Printing Software	All subkeys and values		Opaque data for DEC Networking monitors.
HKLM\ SOFTWARE\ 	All subkeys and values		Opaque data for HP JetAdmin monitors.

Registry key	Value	Type	Description
Hewlett-Packard\ HP JetAdmin			
HKLM\ SOFTWARE\ Lexmark	All subkeys and values		Opaque data for Lexmark Networking monitors.
HKLM\ SOFTWARE\ Microsoft\ Windows NT\ CurrentVersion\ Ports			List of all ports installed with the print server.
	[Port Name]	REG_SZ	Implementation-specific initialization parameters for the port named [Port Name]. If there are no initialization parameters, the value must specify an empty string.
HKLM\ SOFTWARE\ Microsoft\ Windows NT\ CurrentVersion\ Print\ Printers			
	DefaultSpoolDirectory	REG_SZ	The directory used by the print server for storing temporary print job data.
HKLM\ SOFTWARE\ Microsoft\ Windows NT\ CurrentVersion\ Print\ Printers\ [Printer Name]			Information about the printer named [Printer Name].
	Share Name	REG_SZ	Persisted PRINTER_INFO member pShareName .
	Print Processor	REG_SZ	Persisted PRINTER_INFO member pPrintProcessor .
	Datatype	REG_SZ	Persisted PRINTER_INFO member pDatatype .

Registry key	Value	Type	Description
	Parameters	REG_SZ	Persisted PRINTER_INFO member pParameters .
	ObjectGUID		Persisted PRINTER_INFO member pszObjectGUID .
	Printer Driver	REG_SZ	Persisted PRINTER_INFO member pDriverName .
	Default DEVMODE	REG_SZ	Persisted PRINTER_INFO_8 member pDevMode .
	Priority	REG_DWORD	Persisted PRINTER_INFO member Priority .
	Default Priority	REG_DWORD	Persisted PRINTER_INFO member DefaultPriority .
	StartTime	REG_DWORD	Persisted PRINTER_INFO member StartTime .
	UntilTime	REG_DWORD	Persisted PRINTER_INFO member UntilTime .
	Separator File	REG_SZ	Persisted PRINTER_INFO member pSepFile .
	Location	REG_SZ	Persisted PRINTER_INFO member pLocation .
	Attributes	REG_DWORD	Persisted PRINTER_INFO member Attributes .
	Security	REG_BINARY	Persisted PRINTER_INFO member pSecurityDescriptor .
	Port	REG_SZ	Persisted PRINTER_INFO member pPortName .

Definitions for placeholders in the above table:

- [Env Name] is an environment name as specified in [2.2.4.5](#).
- [Version] must be "Version-0", "Version-1", "Version-2" or "Version-3". The numbers 0-3 are described in section [2.2.1.3.1](#), member cVersion.
- [Driver Name] is a driver name as specified in section [2.2.4.6](#).
- [Processor Name] is a print processor name as specified in section [2.2.4.8](#).
- [Monitor Name] is a monitor name as specified in section [2.2.4.12](#).
- [Port Name] is a port name as specified in section [2.2.4.13](#).

- [Printer Name] is a printer name as specified in section [2.2.4.2](#).

<75> [Section 3.1.1](#): Locations of Print System Components in the Server File System

Print system management tools such as **printmig**, **brm**, and other, third-party tools remotely access files that are loaded or executed by the Windows print server. Windows print server loads or executes files as described in the following table.

File description	Location on print server file system
Printer driver files	%systemroot%\system32\spool\drivers\[env-dir]\[version-dir]
Print processor files	%systemroot%\system32\spool\prtprocs\[env-dir]
Separator files	%systemroot%\system32\spool\sepfiles

Definitions for placeholders in the above table:

[env-dir] must specify one of the following strings, depending on the [Env Name] of the driver:

- "W32X86" for the environment string "Windows NT x86"
- "IA64" for the environment string "Windows IA64"
- "WIN40" for the environment string "Windows 4.0"
- "W32ALPHA" for the environment string "Windows NT Alpha_AXP"
- "X64" for the environment string "Windows X64"

[version-dir] must be one of the strings "0", "1", "2", or "3", matching the [DRIVER_INFO](#) member cVersion.

The %systemroot% environment variable must be shared by the print server using the share name "admin\$".

<76> [Section 3.1.4](#):

Opnum	Description
37	Only used locally by Windows, never remotely.
38	Only used locally by Windows, never remotely.
43	Only used locally by Windows, never remotely.
44	Only used locally by Windows, never remotely.
45	Only used locally by Windows, never remotely.
49	Only used locally by Windows, never remotely.
50	Only used locally by Windows, never remotely.
54	Only used locally by Windows, never remotely.
55	Only used locally by Windows, never remotely.

Opnum	Description
57	Only used locally by Windows, never remotely.
63	Only used locally by Windows, never remotely.
64	Only used locally by Windows, never remotely.
68	Only used locally by Windows, never remotely.
74	Only used locally by Windows, never remotely.
75	Only used locally by Windows, never remotely.
76	Only used locally by Windows, never remotely.
83	Only used locally by Windows, never remotely.
90	Only used locally by Windows, never remotely.
91	Only used locally by Windows, never remotely.
92	Only used locally by Windows, never remotely.
93	Only used locally by Windows, never remotely.
94	Only used locally by Windows, never remotely.
95	Only used locally by Windows, never remotely.
98	Only used locally by Windows, never remotely.
99	Only used locally by Windows, never remotely.
100	Only used locally by Windows, never remotely.
101	Only used locally by Windows, never remotely.
102	Only used locally by Windows, never remotely.
103	Only used locally by Windows, never remotely.
104	Only used locally by Windows, never remotely.
105	Only used locally by Windows, never remotely.
106	Only used locally by Windows, never remotely.
107	Only used locally by Windows, never remotely.
108	Only used locally by Windows, never remotely.
109	Only used locally by Windows, never remotely.

<77> [Section 3.1.4.1.2](#): Windows always uses the same server name for the **pName** parameter and to create the **RPC binding**.

IPv6 names are supported only on Windows Server 2008 and Windows Vista.

<78> [Section 3.1.4.1.9.7:](#) If the value of the **pSecurity** member in the **SECURITY_CONTAINER** structure is null, a default security descriptor is used. Security descriptors are specified in [\[MS-DTYP\]](#).

Windows Vista:

The owner in the security descriptor is the local system. The **DACL** contains **ACEs**, which grant the following permissions:

Full control to the administrator's group and to the user who added the printer.

Print permissions to "everyone".

Permissions to control (cancel, pause, resume) the job to the user who submits the job.

Windows XP, Windows Server 2003, Windows 2000, and Windows NT:

The owner in the security descriptor is the user who added the printer. The DACL contains ACEs that grant the following permissions:

Full control to administrators and power users groups.

Print permissions to "everyone".

Permissions to control (cancel, pause, resume) the job to the user who submits the job.

<79> [Section 3.1.4.1.9.8:](#) Windows does not use the following members: **pUserName**, **dwBuildNum**, **dwMajorVersion**, **dwMinorVersion**, and **wProcessorArchitecture**. **pMachineName** is used only if the server cannot determine the client machine name using RPC functions. The **pMachineName** member may be null.

<80> [Section 3.1.4.2.1:](#) Although this value is defined, Windows does not use this value.

<81> [Section 3.1.4.2.1:](#) The server checks that the client user has the **SERVER_ACCESS_ENUMERATE** permission.

<82> [Section 3.1.4.2.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<83> [Section 3.1.4.2.1:](#) Printers are returned in alphabetical order. Windows uses a policy-defined period (default: 60 minutes) from the first call to [RpcAddPrinter](#) with **Level** set to one to determine whether **ERROR_CAN_NOT_COMPLETE**, as specified in [\[MS-ERREF\]](#), is returned.

<84> [Section 3.1.4.2.1:](#) Unshared printers are enumerated only if the user has the **SERVER_ACCESS_ADMINISTER** permission.

Only those printers whose security descriptor grants **PRINTER_ACCESS_USE** to the caller are enumerated.

<85> [Section 3.1.4.2.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<86> [Section 3.1.4.2.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<87> [Section 3.1.4.2.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<88> [Section 3.1.4.2.3](#): The server checks that the client user has implementation-specific permissions to install a printer, typically [SERVER_ACCESS_ADMINISTER](#).

<89> [Section 3.1.4.2.3](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<90> [Section 3.1.4.2.3](#): Windows stores the time when the printer is added to the list. When a printer is added, Windows removes any printer on the list that was added more than 70 minutes ago. Windows stores a maximum of 256 printers in the list. If the limit is reached, no new printers are added, and **ERROR_OUTOFMEMORY**, as specified in [\[MS-ERREF\]](#), is returned.

<91> [Section 3.1.4.2.3](#): Windows fails this call if the server does not have installed all the following: the printer driver, the port, and the print processor.

<92> [Section 3.1.4.2.3](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<93> [Section 3.1.4.2.4](#): The server verifies that the printer object has been opened with an access value including the generic DELETE permission, for example, [PRINTER_ALL_ACCESS](#).

<94> [Section 3.1.4.2.4](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<95> [Section 3.1.4.2.4](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<96> [Section 3.1.4.2.5](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<97> [Section 3.1.4.2.5](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<98> [Section 3.1.4.2.6](#): Windows Vista, Windows Server 2008: This parameter MUST specify a handle to a printer object or server object if the value of the **Level** parameter is three.

<99> [Section 3.1.4.2.6](#): The server verifies that the printer object has been opened with the [PRINTER_ACCESS_USE](#) access values requested.

<100> [Section 3.1.4.2.6](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<101> [Section 3.1.4.2.6](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<102> [Section 3.1.4.2.7](#): The server checks that the client user has [SERVER_ACCESS_ENUMERATE](#) permission.

<103> [Section 3.1.4.2.7](#): Windows returns a non-zero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<104> [Section 3.1.4.2.7](#): For printer objects, the data identified by **pValueName** is stored in the registry under the key "PrinterDriverData"; therefore, [RpcGetPrinterDataEx](#) is used with [pKeyName](#) pointing to the string "PrinterDriverData" to access the identical set of values.

<105> [Section 3.1.4.2.7](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<106> Section 3.1.4.2.8:](#) For print servers, this string is one of the predefined strings listed in [Server Handle Key Values](#), section [2.2.3.8](#).

[<107> Section 3.1.4.2.8:](#) If **hPrinter** is a server object handle, the server checks that the client user has [SERVER_ACCESS_ADMINISTER](#) permission; if **hPrinter** is a printer object handle, the server checks that the client user has [PRINTER_ACCESS_ADMINISTER](#) permission.

[<108> Section 3.1.4.2.8:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<109> Section 3.1.4.2.8:](#) For printer objects, the data identified by **pValueName** is stored in the registry under the key named "PrinterDriverData"; therefore, [RpcSetPrinterData](#) and [RpcSetPrinterDataEx](#) are used with [pKeyName](#) pointing to the string "PrinterDriverData" alias to an identical set of values.

[<110> Section 3.1.4.2.8:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<111> Section 3.1.4.2.9:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<112> Section 3.1.4.2.9:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<113> Section 3.1.4.2.10:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<114> Section 3.1.4.2.10:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<115> Section 3.1.4.2.11:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<116> Section 3.1.4.2.11:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<117> Section 3.1.4.2.12:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<118> Section 3.1.4.2.12:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<119> Section 3.1.4.2.13:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<120> Section 3.1.4.2.13:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<121> Section 3.1.4.2.14:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<122> Section 3.1.4.2.14:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<123> Section 3.1.4.2.15:](#) The server checks that the client user has implementation-specific permissions to install a printer, typically [SERVER_ACCESS_ADMINISTER](#).

<124> [Section 3.1.4.2.15](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<125> [Section 3.1.4.2.15](#): Windows stores the time when the printer is added to the list. When a printer is added, Windows removes any printer on the list that was added more than a policy-defined period (default 70 minutes) ago. Windows stores a maximum of 256 printers in the list. If the limit is reached, no new printers are added, and ERROR_OUTOFMEMORY is returned, as specified in [\[MS-ERREF\]](#).

<126> [Section 3.1.4.2.15](#): Windows fails this call if, at the time of this call, the server does not have installed all the following: the printer driver, the port, and the print processor.

<127> [Section 3.1.4.2.15](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<128> [Section 3.1.4.2.16](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<129> [Section 3.1.4.2.16](#): For printer objects, the data identified by **pValueName** is stored in the registry under the key named "PrinterDriverData"; therefore, [RpcEnumPrinterData](#) and [RpcEnumPrinterDataEx](#) used with [pKeyName](#) pointing to the string "PrinterDriverData" alias to an identical set of values.

<130> [Section 3.1.4.2.16](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<131> [Section 3.1.4.2.17](#): The server checks that the client user has [PRINTER_ACCESS_ADMINISTER](#) permission.

<132> [Section 3.1.4.2.17](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<133> [Section 3.1.4.2.17](#): For printer objects, the data identified by **pValueName** is stored in the registry under the key named "PrinterDriverData"; therefore, [RpcDeletePrinterData](#) and [RpcDeletePrinterDataEx](#) used with [pKeyName](#) pointing to the string "PrinterDriverData" alias to an identical set of values.

<134> [Section 3.1.4.2.17](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<135> [Section 3.1.4.2.18](#): If **hPrinter** is a server object handle, the server checks that the client user has [SERVER_ACCESS_ADMINISTER](#) permission. If **hPrinter** is a printer object handle, the server checks that the client user has [PRINTER_ACCESS_ADMINISTER](#) permission.

<136> [Section 3.1.4.2.18](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<137> [Section 3.1.4.2.18](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<138> [Section 3.1.4.2.19](#): The server checks that the client user has [SERVER_ACCESS_ENUMERATE](#) permission.

<139> [Section 3.1.4.2.19](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<140> [Section 3.1.4.2.19:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<141> [Section 3.1.4.2.20:](#) Windows returns a non-zero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<142> [Section 3.1.4.2.20:](#) Windows returns a non-zero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<143> [Section 3.1.4.2.21:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<144> [Section 3.1.4.2.21:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<145> [Section 3.1.4.2.22:](#) The server checks that the client user has the [PRINTER_ACCESS_ADMINISTER](#) permission.

<146> [Section 3.1.4.2.22:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<147> [Section 3.1.4.2.22:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<148> [Section 3.1.4.2.23:](#) The server checks that the client user has the [PRINTER_ACCESS_ADMINISTER](#) permission.

<149> [Section 3.1.4.2.23:](#) Windows returns a nonzero **Windows error code** to indicate failure, as specified in [\[MS-ERREF\]](#).

<150> [Section 3.1.4.2.23:](#) Windows returns a nonzero **Windows error code** to indicate failure, as specified in [\[MS-ERREF\]](#).

<151> [Section 3.1.4.2.24:](#) The name of the provider file is used and is stored in the Windows registry. If **pProvider** is null, the Windows operating system uses Win32spl.dll as the name of the executable object.

<152> [Section 3.1.4.2.24:](#) The server checks that the client user has the [SERVER_ACCESS_ADMINISTER](#) permission.

<153> [Section 3.1.4.2.24:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<154> [Section 3.1.4.2.24:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<155> [Section 3.1.4.2.25:](#) The server checks that the client user has the [SERVER_ACCESS_ADMINISTER](#) permission.

<156> [Section 3.1.4.2.25:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<157> [Section 3.1.4.2.25:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<158> [Section 3.1.4.2.26:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<159> Section 3.1.4.2.26:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<160> Section 3.1.4.2.27:](#) The Windows system port monitors do not support any additional action strings.

[<161> Section 3.1.4.2.27:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<162> Section 3.1.4.2.27:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<163> Section 3.1.4.3.1:](#) The server verifies that the printer object has been opened with one of the following access values requested, depending on the type of operation:

- [PRINTER_ACCESS_ADMINISTER](#) for changing job position.
- Generic DELETE permission for [JOB_CONTROL_CANCEL](#) and [JOB_CONTROL_DELETE](#).
- [JOB_ACCESS_ADMINISTER](#) for all others.

[<164> Section 3.1.4.3.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<165> Section 3.1.4.3.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<166> Section 3.1.4.3.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<167> Section 3.1.4.3.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<168> Section 3.1.4.3.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<169> Section 3.1.4.3.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<170> Section 3.1.4.3.4:](#) Windows servers return **ERROR_INVALID_PARAMETER**.

[<171> Section 3.1.4.3.4:](#) The server verifies that the printer object has been opened with one of the following access values requested: [PRINTER_ACCESS_ADMINISTER](#), [PRINTER_ACCESS_USE](#), [PRINTER_ALL_ACCESS](#), [PRINTER_READ](#), [PRINTER_WRITE](#), or [PRINTER_EXECUTE](#).

[<172> Section 3.1.4.3.4:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<173> Section 3.1.4.3.4:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<174> Section 3.1.4.3.5:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<175> Section 3.1.4.3.5:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<176> Section 3.1.4.4.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<177> Section 3.1.4.4.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<178> Section 3.1.4.4.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<179> Section 3.1.4.4.2:](#) If the string contains "All" or "AllCluster", all printer drivers for all environments must be enumerated.

[<180> Section 3.1.4.4.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<181> Section 3.1.4.4.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<182> Section 3.1.4.4.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<183> Section 3.1.4.4.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<184> Section 3.1.4.4.4:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<185> Section 3.1.4.4.4:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<186> Section 3.1.4.4.5:](#) The server checks that the client user has the [SERVER_ACCESS_ADMINISTER](#) permission.

[<187> Section 3.1.4.4.5:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<188> Section 3.1.4.4.5:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<189> Section 3.1.4.4.6:](#) The following table shows the unsigned 32-bit major operating system version number that is used by Windows clients and servers :

Major version	Meaning
0x00000004	The operating system is Windows Me, Windows 98, Windows 95, or Windows NT 4.0.
0x00000005	The operating system is Windows Server 2003, Windows Server 2003 R2, Windows XP, or Windows 2000.
0x00000006	The operating system is Windows Server 2008 or Windows Vista.

[<190> Section 3.1.4.4.6:](#) The following table shows the unsigned 32-bit minor operating system version number that is used by Windows Server 2008 clients and servers:

Minor version	Meaning
0x00000000	The operating system is Windows Server 2008 or Windows Vista, Windows 2000, Windows 95, or Windows NT 4.0.
0x00000001	The operating system is Windows XP.
0x00000002	The operating system is Windows XP Professional x64 Edition, Windows Server 2003, or Windows Server 2003 R2.
0x0000000A	The operating system is Windows 98.
0x0000005A	The operating system is Windows Me.

<191> [Section 3.1.4.4.6:](#) **pdwServerMaxVersion** and **pdwServerMinVersion** are ignored by the Windows print server, and therefore no values are returned.

Note If both **dwClientMajorVersion** and **dwClientMinorVersion** are set to 0xFFFFFFFF, the print server will return printer driver information for the printer driver version matching the operating system version on which the print server is running.

<192> [Section 3.1.4.4.6:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<193> [Section 3.1.4.4.6:](#) Windows does not do this.

<194> [Section 3.1.4.4.6:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<195> [Section 3.1.4.4.7:](#) Printer Driver Version

This parameter is one of the following values:

Value	Meaning
0x00000001	Windows NT 3.51 user-mode printer drivers
0x00000002	Windows NT 4.0 user-mode printer drivers
0x00000003	Windows 2000 and later user-mode printer drivers

<196> [Section 3.1.4.4.7:](#) The server checks that the client user has the [SERVER ACCESS ADMINISTER](#) permission.

<197> [Section 3.1.4.4.7:](#) Windows returns a nonzero **Windows error code** to indicate failure, as specified in [\[MS-ERREF\]](#).

<198> [Section 3.1.4.4.7:](#) Windows returns a nonzero **Windows error code** to indicate failure, as specified in [\[MS-ERREF\]](#).

<199> [Section 3.1.4.4.8:](#) When adding a printer driver to a print server cluster, do not copy the driver files to the shared cluster disk.

<200> [Section 3.1.4.4.8:](#) Add the printer driver to cluster spooler servers.

[<201> Section 3.1.4.4.8:](#) If the printer driver is blocked from installation by server policy: return ERROR_PRINTER_DRIVER_BLOCKED if this bit is set; if this bit is not set, return ERROR_UNKNOWN_PRINTER_DRIVER.

[<202> Section 3.1.4.4.8:](#) The server checks that the client user has the [SERVER_ACCESS_ADMINISTER](#) permission.

[<203> Section 3.1.4.4.8:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<204> Section 3.1.4.4.8:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<205> Section 3.1.4.5.1:](#) The server checks that the client user has [SERVER_ACCESS_ADMINISTER](#) permission.

[<206> Section 3.1.4.5.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<207> Section 3.1.4.5.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<208> Section 3.1.4.5.2:](#) The server checks that the client user has [SERVER_ACCESS_ADMINISTER](#) permission.

[<209> Section 3.1.4.5.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<210> Section 3.1.4.5.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<211> Section 3.1.4.5.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<212> Section 3.1.4.5.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<213> Section 3.1.4.5.4:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<214> Section 3.1.4.5.4:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<215> Section 3.1.4.5.5:](#) The server checks that the client user has [SERVER_ACCESS_ENUMERATE](#) permission.

[<216> Section 3.1.4.5.5:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<217> Section 3.1.4.5.5:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<218> Section 3.1.4.6.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<219> Section 3.1.4.6.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<220> [Section 3.1.4.6.2](#): Windows Server 2008 and Windows Vista check that the client user has the [SERVER_ACCESS_ENUMERATE](#). All other Windows Vista versions check that the client user has the [SERVER_ACCESS_ADMINISTER](#).

<221> [Section 3.1.4.6.2](#): Windows Vista returns a nonzero Windows Vista error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<222> [Section 3.1.4.6.2](#): Windows Vista returns a nonzero Windows Vista error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<223> [Section 3.1.4.6.3](#): The server checks that the client user has the [SERVER_ACCESS_ADMINISTER](#) permission.

<224> [Section 3.1.4.6.3](#): Windows Vista returns a nonzero Windows Vista error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<225> [Section 3.1.4.6.3](#): Windows Vista returns a nonzero Windows Vista error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<226> [Section 3.1.4.6.4](#): The server checks that the client user has [SERVER_ACCESS_ADMINISTER](#) permission.

<227> [Section 3.1.4.6.4](#): Windows Vista returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<228> [Section 3.1.4.6.4](#): Windows Vista returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<229> [Section 3.1.4.6.5](#): The actions and data retrieval supported by [RpcXcvData](#) are performed by executing the XcvData method on the port monitor module referenced by the input handle. For more information about monitor module methods, see section [3.1.4.11](#).

<230> [Section 3.1.4.6.5](#): For Windows-specific behavior, see section [3.1.4.11](#).

<231> [Section 3.1.4.6.5](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<232> [Section 3.1.4.6.5](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<233> [Section 3.1.4.7.1](#): The server checks that the client user has [SERVER_ACCESS_ENUMERATE](#) permission.

<234> [Section 3.1.4.7.1](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<235> [Section 3.1.4.7.1](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<236> [Section 3.1.4.7.2](#): The server checks that the client user has [SERVER_ACCESS_ADMINISTER](#) permission.

<237> [Section 3.1.4.7.2](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<238> [Section 3.1.4.7.2](#): Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<239> Section 3.1.4.7.3:](#) The server checks that the client user has [SERVER_ACCESS_ADMINISTER](#) permission.

[<240> Section 3.1.4.7.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<241> Section 3.1.4.7.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<242> Section 3.1.4.8.1:](#) The server checks that the client user has [SERVER_ACCESS_ADMINISTER](#) permission.

[<243> Section 3.1.4.8.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<244> Section 3.1.4.8.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<245> Section 3.1.4.8.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<246> Section 3.1.4.8.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<247> Section 3.1.4.8.3:](#) The server checks that the client user has [SERVER_ACCESS_ENUMERATE](#) permission.

[<248> Section 3.1.4.8.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<249> Section 3.1.4.8.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<250> Section 3.1.4.8.4:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<251> Section 3.1.4.8.4:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<252> Section 3.1.4.8.5:](#) The server checks that the client user has [SERVER_ACCESS_ENUMERATE](#) permission.

[<253> Section 3.1.4.8.5:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<254> Section 3.1.4.8.5:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<255> Section 3.1.4.9.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<256> Section 3.1.4.9.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<257> Section 3.1.4.9.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<258> Section 3.1.4.9.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<259> Section 3.1.4.9.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<260> Section 3.1.4.9.4:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<261> Section 3.1.4.9.4:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<262> Section 3.1.4.9.5:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<263> Section 3.1.4.9.5:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<264> Section 3.1.4.9.6:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<265> Section 3.1.4.9.6:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<266> Section 3.1.4.9.7:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<267> Section 3.1.4.9.7:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<268> Section 3.1.4.9.8:](#) Windows returns a nonzero Windows error, as specified in [\[MS-ERREF\]](#).

[<269> Section 3.1.4.9.8:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<270> Section 3.1.4.10.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<271> Section 3.1.4.10.1:](#) Windows waits for up to 600 seconds

[<272> Section 3.1.4.10.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<273> Section 3.1.4.10.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<274> Section 3.1.4.10.3:](#) This method is called only by Windows NT 3.5 clients.

[<275> Section 3.1.4.10.3:](#) Windows clients generate an arbitrary unique value for each known printer.

[<276> Section 3.1.4.10.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

[<277> Section 3.1.4.10.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<278> [Section 3.1.4.10.4:](#) Windows clients generate an arbitrary unique value for each known printer.

<279> [Section 3.1.4.10.4:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<280> [Section 3.1.4.10.4:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<281> [Section 3.1.4.10.5:](#) Windows clients pass in a monotonically increasing value for **dwColor** in the [RpcRouterRefreshPrinterChangeNotification](#) call. Windows servers use this value to set the **dwColor** parameter in the [RpcRouterReplyPrinterEx](#) call. Windows clients determine the most recent notification data that the server returns by comparing the values of **dwColor**. Windows clients discard the notification data received on the [RpcRouterReplyPrinterEx](#) call if **dwColor** is different from **dwColor** received on the most recent [RpcRouterRefreshPrinterChangeNotification](#) call. This mechanism is required because of the possible ordering reversal of calls, caused by network latency.

<282> [Section 3.1.4.10.5:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<283> [Section 3.1.4.10.5:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<284> [Section 3.1.4.11:](#) The Windows operating system provides port monitor modules in the form of executable dynamic-link libraries (DLLs). Examples are:

- **localmon.dll**, which manages local serial ("COM") and parallel ("LPT") ports on a Windows machine (see section below).
- **lprmon.dll**, which allows a Windows print server to send print jobs to machines that support Unix print-server functions (see section below).
- **tcpmon.dll**, which manages standard TCP/IP ports on a Windows machine.
- **wsdmon.dll**, which manages Web Services for Devices ports (Windows Server 2008 and Windows Vista only).

In Windows 2000, Windows Me, Windows XP, Windows Server 2003, Windows Vista and Windows Server 2008 all the functions that **localmon** exports were incorporated into **localspl.dll**. Another change in those versions of Windows is that port monitors are divided into two DLLs, a port monitor server DLL and a port monitor user interface (UI) DLL.

The following sections, [Localmon](#) and [Lprmon](#), describe the implementation of the **XcvData** method, its supported actions and corresponding behaviors in the localmon and lprmon monitor modules.

All method descriptions assume the standard buffer size validation pattern, as specified for the [RpcXcvData](#) method in section [3.1.4.6.5](#). Unless otherwise specified, for actions not using **pInputData**, **pInputData** should be null and **cbInputData** should be zero. Unless otherwise specified, for actions not using **pOutputData**, **pOutputData** should be null, **cbOutputData** should be zero, and **pcbOutputNeeded** should be null.

<285> [Section 3.1.4.11.1:](#) This section describes the implementation of **XcvData** methods in **localmon**. This monitor module is used by Windows to control parallel and serial ports that may have a printer connected to them.

Starting with Windows 2000, localmon is packaged inside another executable component called **localspl.dll**; the presentation is captured in a separate monitor module. The user interface module corresponding to localmon is called localui.

For parallel and serial port naming, see section [2.2.4.13](#). For more information about the values listed in the left column in the following table, see section [3.1.4.6.5](#).

Value	Meaning
"MonitorUI"	This action returns the name of the client-side monitor DLL. The pOutputData parameter must be a non-null pointer to the buffer that receives the string representing the name of the DLL.
"AddPort"	This action adds a local port. The pInputData parameter must be a non-null pointer to a string representing the name of the port.
"DeletePort"	This action deletes a local port. The pInputData parameter must be a non-null pointer to a string representing the name of the port.
"ConfigureLPTPortCommandOK"	This action determines configuration of the LPT port and sets the transmission retry timeout. The pInputData parameter must be a non-null pointer to a string representing a number from one through three inclusive. Those numbers map to the port to configure; that is, whether it is "LPT1" OR "LPT2" OR "LPT3". The pOutputData parameter is not used.
"PortExists"	This action checks whether a local port exists in the print server's list of port objects. The pInputData parameter must be a non-null pointer to a string representing the name of the port. The pOutputData parameter must be a non-null pointer to a DWORD variable that must receive the value 0 if the port does not exist and a nonzero value if the port exists.
"PortIsValid"	This action determines whether a given name is a valid port name accepted by the port monitor. The pInputData parameter must be a non-null pointer to a string representing the name of the port. If the port identifies a valid port name, the method returns ERROR_SUCCESS ; otherwise, it returns a nonzero error code.

[<286> Section 3.1.4.11.2:](#) This section describes the implementation of **XcvData** methods in lprmon. This monitor module is used by Windows to control printers over a network on machines that have implemented Unix print server functions and expose them through the **Line Printer (LPR) Protocol**, as described in [RFC1179]. This monitor module is supported in versions beginning with Windows 2000. The user interface module corresponding to lprmon is called **lprmonui**.

For network port naming, see section [2.2.4.13](#). For more information about the values listed in the Value column in the following table, see section [3.1.4.6.5](#).

Value	Meaning
"MonitorUI"	This action returns the name of the client-side monitor DLL. The pOutputData parameter must be a non-null pointer to a buffer that must receive the string representing the name of the DLL on completion of the action that the pOutputData parameter must contain.
"AddPort"	This action adds an LPR printer port. The pInputData parameter must be a non-null pointer to a string representing the name of the port. The pOutputData parameter is not used.

Value	Meaning
"DeletePort"	This action deletes an LPR printer port. The pInputData parameter must be a non-null pointer to a string representing the name of the port. The pOutputData parameter is not used.
"CheckPrinter"	This action checks on the LPR printer port. The pInputData parameter must be a non-null pointer to a string representing the name of the port connected to the printer.

<287> [Section 3.1.4.11.3](#): The tcpmon monitor module is used by Windows to control printers directly connected to a **TCP/IP** network. This monitor module is supported starting with Windows 2000. The user interface module corresponding to tcpmon is called **tcpmonui**.

For network port naming, see section [2.2.4.13](#).

<288> [Section 3.1.4.11.3.1.1](#):

```
typedef struct _PORT_DATA_1 {
    WCHAR    sztPortName[64];
    DWORD    dwVersion;
    DWORD    dwProtocol;
    DWORD    cbSize;
    DWORD    dwReserved;
    WCHAR    sztHostAddress[49];
    WCHAR    sztSNMPCommunity[33];
    DWORD    dwDoubleSpool;
    WCHAR    sztQueue[33];
    WCHAR    sztIPAddress[16];
    WCHAR    sztHardwareAddress[13];
    WCHAR    sztDeviceType[257];
    DWORD    dwPortNumber;
    DWORD    dwSNMPEnabled;
    DWORD    dwSNMPDevIndex;
} PORT_DATA_1, *PPORT_DATA_1;
```

szPortName: Specifies the name of the port.

dwVersion: Specifies the version number of the PORT_DATA_1 structure, which is currently 1.

dwProtocol: Specifies the protocol to use for the port. This value can be either PROTOCOL_RAWTCP_TYPE (1), indicating that the port expects RAW print data, or PROTOCOL_LPR_TYPE (2), indicating that the port expects to be driven as an LPR port.

cbSize: Specifies the size, in bytes of this structure. Use sizeof(PORT_DATA_1) for this value.

dwReserved: Reserved, must be set to zero.

szHostAddress: Specifies the IP Address or host name of the printer.

szSNMPCommunity: Specifies the SNMP community name of the printer.

dwDoubleSpool: If nonzero, indicates that double spooling is enabled. If zero, double spooling is disabled.

szQueue: Specifies the LPR queue name.

szIPAddress: Specifies the IPv4 address of the printer.

szHardwareAddress: Specifies the MAC address of the printer.

szDeviceType: Generic SNMP device description (OID 1.3.6.1.2.1.1.1).

dwPortNumber: Specifies the port number of the device.

dwSNMPEnabled: If TRUE, indicates that the device supports Simple Network Management Protocol (SNMP).

dwSNMPDevIndex: Specifies the SNMP device index.

[<289> Section 3.1.4.11.3.1.2:](#)

```
typedef struct _DELETE_PORT_DATA_1 {
    WCHAR    psztPortName[64];
    WCHAR    psztName[49];
    DWORD    dwVersion;
    DWORD    dwReserved;
} DELETE_PORT_DATA_1, *PDELETE_PORT_DATA_1;
```

psztPortName: Specifies the name of the port to be deleted.

psztName: Specifies the server name for the port to be deleted.

dwVersion: Specifies the version of this structure, which is currently 1.

dwReserved: Is obsolete, and must be set to 0.

[<290> Section 3.1.4.11.3.1.3:](#)

```
typedef struct _CONFIG_INFO_DATA_1 {
    WCHAR    szPortName[64];
    DWORD    dwVersion;
} CONFIG_INFO_DATA_1, *PCONFIG_INFO_DATA_1;
```

szPortName: Specifies the name of the port to be queried for configuration information.

dwVersion: Specifies the level of the PORT_DATA structure that will contain the configuration information. This value must be one or two.

[<291> Section 3.1.4.11.3.1.4:](#)

```
typedef struct _PORT_DATA_LIST_1
{
    DWORD    dwVersion;
    DWORD    cPortData;
    PORT_DATA_2 pPortData[];
} PORT_DATA_LIST_1, *PPORT_DATA_LIST_1;
```

dwVersion: Specifies the version number of the [PORT_DATA_LIST_1](#) structure, which is currently 1.

cPortData: The number of [PORT_DATA_2](#) structures contained in the **pPortData** array of this [PORT_DATA_LIST_1](#) structure.

[<292> Section 3.1.4.11.3.1.5:](#)

```
typedef struct _PORT_DATA_2
{
    WCHAR    sztPortName[64];
    DWORD    dwVersion;
    DWORD    dwProtocol;
    DWORD    cbSize;
    DWORD    dwReserved;
    WCHAR    sztHostAddress [128];
    WCHAR    sztSNMPCommunity[33];
    DWORD    dwDoubleSpool;
    WCHAR    sztQueue[33];
    WCHAR    sztDeviceType[257];
    DWORD    dwPortNumber;
    DWORD    dwSNMPEnabled;
    DWORD    dwSNMPDevIndex;
    DWORD    dwPortMonitorMibIndex;
} PORT_DATA_2, *PPORT_DATA_2;
```

szPortName: Specifies the name of the port.

dwVersion: Specifies the version number of the [PORT_DATA_2](#) structure, which is currently 1.

dwProtocol: Specifies the protocol to use for the port. This value can be either `PROTOCOL_RAWTCP_TYPE` or `PROTOCOL_LPR_TYPE`. For more information, see [PORT_DATA_2](#).

cbSize: Specifies the size, in bytes, of this structure. Use `sizeof(PORT_DATA_2)` for this value.

dwReserved: Reserved, must be set to zero.

szHostAddress: Specifies the IP Address or host name of the printer.

sztSNMPCommunity: Specifies the SNMP community name of the printer.

dwDoubleSpool: If `TRUE`, indicates that double spooling is enabled. If `FALSE`, double spooling is disabled.

szQueue: Specifies the LPR queue name.

szDeviceType: Generic SNMP device description (OID 1.3.6.1.2.1.1.1).

dwPortNumber: Specifies the port number of the device.

dwSNMPEnabled: If nonzero, indicates that the device supports Simple Network Management Protocol (SNMP).

dwSNMPDevIndex: Specifies the SNMP device index.

dwPortMonitorMibIndex: Specifies the Index in the network devices PWG Port Monitor MIB for the current tcpmon port. This index is used to query the IEEE 1284 device ID for the attached printer. For more information, see [\[IEEE1284\]](#).

<293> [Section 3.1.4.11.3.2:](#) For more information about the values listed in the left column of the following table, see section [3.1.4.6.5](#).

Value	Meaning
"AddPort"	This action adds a printer port. The pInputData parameter must be a non-null pointer to a PORT_DATA_1 or PORT_DATA_2 structure.
"DeletePort"	This action deletes a printer port. The pInputData parameter must be a non-null pointer to a DELETE_PORT_DATA_1 structure.
"MonitorUI"	This action returns the name of the client-side monitor DLL. The pOutputData parameter must be a non-null pointer to a buffer that must receive the string representing the name of the DLL.
"ConfigPort"	This action configures a printer port. The pInputData parameter must be a non-null pointer to a PORT_DATA_1 or PORT_DATA_2 structure.
"GetConfigInfo"	This action gets configuration information for a printer port. The pInputData parameter must be a non-null pointer to a CONFIG_INFO_DATA_1 structure. The pOutputData parameter must point to a buffer that must receive a PORT_DATA_1 or PORT_DATA_2 structure describing the port.
"HostAddress"	This action gets the printer's host name. pOutputData must be a non-null pointer to a buffer that must receive a string containing the printer's host name.
"IPAddress"	This action gets the printer's IP address. pOutputData must be a non-null pointer to a buffer that must receive a string containing the printer's IP address.
"SNMPCommunity"	This action gets the printer's SNMP community name. pOutputData must be a non-null pointer to a buffer that must receive a string containing the printer's SNMP community name.
"SNMPDeviceIndex"	This action gets the printer's SNMP device index. pOutputData must be a non-null pointer to a variable that must receive a DWORD value containing the printer's SNMP device index.
"SNMPEnabled"	This action determines whether SNMP is enabled for the printer. pOutputData must be a non-null pointer to a variable that must receive the DWORD value 0 if SNMP is disabled and a nonzero value otherwise.
"GetIdlePollingState"	This action determines whether tcpmon is set to poll automatically for the printer ("idle polling"). pOutputData must be a non-null pointer to a variable that must receive the DWORD value 0 if idle polling is disabled, and a nonzero value otherwise.
"SetIdlePollingState"	This action sets the idle polling state for the printer. pInputData must be a non-null pointer to a DWORD which must be set to 1 to enable idle polling, and to 0 to disable it.
"SetDeviceIDOID"	This action sets the OID used to query the IEEE 1284 device ID from the printer. (For more information, see [IEEE1284] .) pInputData must be a non-null pointer to a string specifying the OID. If this OID is not set, tcpmon uses a default of "1.3.6.1.4.1.2699.1.2.1.2.1.1.3.<1-based port index>".
"DeviceID"	This action initiates a query for the IEEE 1284 device ID. (For more information, see [IEEE1284] .) pOutputData must be a non-null pointer to a buffer that must receive a string containing the IEEE 1284 device ID, as specified in [IEEE1284] .

Value	Meaning
"GetPortList"	This action requests the list of supported ports on a device. pInputData must be a non-null pointer to a string containing the IP address of hostname of the device. pOutputData must be a non-null pointer to a buffer that must receive a PORT_DATA_LIST_1 structure.
"CleanupPort"	This action attempts to remove the TCP/IP port associated with the hXcv handle. If printers are still using the port, it will not be removed.

<294> [Section 3.1.4.11.4:](#) This section describes the implementation of the XcvData method in wsdmon. This monitor module is used by Windows to control Web Services for Devices printers. This monitor module is supported only on Windows Server 2008 and Windows Vista. The wsdmon does not have a corresponding user interface module.

For network port naming, see section [2.2.4.13](#).

<295> [Section 3.1.4.11.4.1.1:](#)

```
typedef struct _WSD_DRIVER_DATA
{
    WCHAR    szIEEE1284Id[260];
    WCHAR    szModelName[260];
} WSD_DRIVER_DATA, *PWSD_DRIVER_DATA;
```

szIEEE1284Id: Specifies the IEEE 1284 device ID for the discovered printer. (For more information, see [IEEE1284](#).) This can be used to generate the correct Windows **PnP ID** for the printer driver.

szModelName: Specifies the name of the printer model discovered.

<296> [Section 3.1.4.11.4.1.2:](#)

```
enum DISCOVERY_METHOD
{
    kMulticast,
    kDirected
};

typedef struct _WSD_BACKUP_PORT_DATA
{
    DISCOVERY_METHOD    DiscoveryMethod;
    WCHAR                szGlobalID[1024];
    WCHAR                szRemoteURL[1024];
} WSD_BACKUP_PORT_DATA, *PWSD_BACKUP_PORT_DATA;
```

DiscoveryMethod: specifies how the WSD port was initially discovered.

This value MUST be **kMulticast** if the WSD was initially discovered by using multicast discovery. This value MUST be **kDirected** if the WSD was initially discovered by using directed discovery.

szGlobalID: If **DiscoveryMethod** is **kMulticast**, this value specifies the **PKEY_PNPX_GlobalIdentify** of the device attached to the WSD port; otherwise, it MUST be empty.

szRemoteURL: If **DiscoveryMethod** is **kDirected**, this value specifies the **URL** of the device attached to the WSD port; otherwise, it MUST be empty.

<297> [Section 3.1.4.11.4.2:](#)

Value	Meaning
"CleanupPort"	This action attempts to remove the WSD port associated with the hXcv handle. If printers are still using the port, it will not be removed.
"DeviceID"	This action initiates a query for the WSD DeviceID . The pOutputData parameter MUST be a non-null pointer to a buffer that MUST receive a string containing the WSD DeviceID (PKEY_PNPX_GlobalIdentity).
"PnPXID"	This action initiates a query for the WSD PnPXID. The pOutputData parameter MUST be a non-null pointer to a buffer that MUST receive a string containing the WSD PnPXID (PKEY_PNPX_ID).
"ResetCommunication"	The ResetCommunication command attempts to make sure that the communication between the printer and the operating system is working properly.
"ServiceID"	This action initiates a query for the WSD ServiceID. The pOutputData parameter MUST be a non-null pointer to a buffer that MUST receive a string containing the WSD ServiceID (PKEY_PNPX_ServiceID).
"CheckCluster"	This action determines whether the queried server is a stand-alone server or a cluster node. The pOutputData parameter MUST be a non-null pointer to a variable that MUST receive the DWORD value 0 if the queried server is not a cluster node and a nonzero value otherwise.
"DiscoverDevice"	This action uses WS-discovery directed unicast search to try to find a WSD enabled device at the supplied URI of the endpoint. The pInputData parameter MUST be a non-null pointer to a string specifying the URI of the WSD endpoint. If a WSD device is found, and it supports the WSD print service definition, ERROR_SUCCESS is returned; otherwise, ERROR_PRINTER_NOT_FOUND is returned, as specified in [MS-ERREF] .
"DriverAvailable"	This action determines whether a printer driver for the queried device is available in the server's driver store. The pInputData parameter MUST be a non-null pointer to a string specifying the URI of the WSD endpoint. The pOutputData parameter MUST be a non-null pointer to a buffer that MUST receive a WSD_DRIVER_DATA structure if the specified endpoint supports a WSD Printer Service and a driver is available. If the endpoint does not support the WSD Printer Service, ERROR_PRINTER_NOT_FOUND is returned, as specified in [MS-ERREF] . If no driver can be found, ERROR_CANNOT_DETECT_DRIVER_FAILURE is returned.
"AssocDevice"	Only supported on stand-alone servers, this action searches for a WSD Printer Service at the supplied URI of the endpoint and installs the printer if found. The pInputData parameter MUST be a non-null pointer to a string specifying the URI of the WSD endpoint. If a WSD Printer Service is found, a PnPX installation of the printer is initiated; otherwise, ERROR_PRINTER_NOT_FOUND is returned, as specified in [MS-ERREF] .

Value	Meaning
"AddPrinterPort"	Only supported on cluster servers, this action searches for a WSD Printer Service at the supplied URI of the endpoint, and if one is found, creates a new WSD Port connected to the discovered device. The pInputData parameter MUST be a non-null pointer to a string specifying the URI of the WSD endpoint. The pOutputData parameter MUST be a non-null pointer to a buffer that MUST receive a string identifying the new port name if a WSD Printer Service is found; otherwise, ERROR_PRINTER_NOT_FOUND is returned, as specified in [MS-ERREF] .
"BackupPort"	This action initiates a query for the WSD port backup data. The pOutputData parameter MUST be a non-null pointer to a buffer that MUST receive a WSD_BACKUP_PORT_DATA structure.
"AssocDeviceMulticast"	Only supported on stand-alone servers, this action searches for a WSD Printer Service at the device endpoint specified by the GlobalID specified by the string pointed to by pInputData and installs the printer if found. The pInputData parameter MUST be a non-null pointer to a string specifying the GlobalID of the WSD endpoint. If a WSD Printer Service is found, a PnP installation of the printer is initiated; otherwise, ERROR_PRINTER_NOT_FOUND is returned, as specified in [MS-ERREF] .

<298> [Section 3.2.4:](#) The Windows implementation ignores errors and passes them back to the invoker.

<299> [Section 3.2.4.1.1:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<300> [Section 3.2.4.1.1:](#) The Windows server calls this method during processing of [RpcRemoteFindFirstPrinterChangeNotificationEx](#). After the [RpcReplyOpenPrinter](#) call returns, the Windows server uses the return value of [RpcReplyOpenPrinter](#) to complete the [RpcRemoteFindFirstPrinterChangeNotificationEx](#) call to indicate the success based on the return value of the [RpcReplyOpenPrinter](#) call.

<301> [Section 3.2.4.1.2:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<302> [Section 3.2.4.1.3:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<303> [Section 3.2.4.1.4:](#) Windows returns a nonzero Windows error code to indicate failure, as specified in [\[MS-ERREF\]](#).

<304> [Section 3.2.4.2.2:](#) Windows servers do not implement this method and return **ERROR_INVALID_PARAMETER**. Windows clients call this method, and upon receiving **ERROR_INVALID_PARAMETER** print documents as described in Printing a Document using [RpcStartDocPrinter](#), section [3.2.4.2.1](#). Windows clients have not been tested with servers supporting [RpcAddJob](#), and it is recommended for interoperability that servers do not implement this method.

<305> [Section 3.2.4.2.4:](#) The Windows system will select a number of jobs to be 0xFFFFFFFF to obtain the full list in a single operation.

<306> [Section 3.2.4.2.6:](#) Windows clients only enumerate their shared printers if no directory services are available in the domain.

[<307> Section 3.2.4.2.6:](#) Windows clients enumerate their shared printers up to a default of three print servers and a default of two workstations and one additional server per 32 print servers found. The values can be changed via policy settings.

[<308> Section 3.2.4.2.7:](#) A Windows client uses [RpcOpenPrinterEx](#) to open a handle for the server object. Then the Windows client uses [RpcGetPrinterData](#) with the value names "Architecture", "MajorVersion", and "OSVersion" to obtain information about the server's environment. This information is used to select an appropriate printer driver. The client subsequently closes the server object handle using [RpcClosePrinter](#). The Windows-based client also uses [RpcEnumPorts](#) to populate a list with ports available on the server from which the end-user can select the port for the new printer. The Windows-based client uses [RpcEnumMonitors](#) to determine a port monitor for the new printer. The Windows-based client uses [RpcGetPrinterDriverDirectory](#) to determine the destination directory in case a printer driver needs to be installed for the new printer.

[<309> Section 5.2:](#) Windows print server follows a security model in which the print server, print queues, and print job are securable resources. Each of these resources has a security descriptor (SD) that is associated with it. The SD contains the security information that is associated with a resource on the print server. The print server checks the client access to resources by comparing the security information that is associated with the caller against the SD of the resource. SDs are specified in [\[MS-DTYP\]](#).

Each RPC client has associated with it an access token, which contains the **security identifier (SID)** of the user making the RPC call. The SD identifies the owner of the printing resource and contains a discretionary access control list (DACL). The DACL contains access control entries (ACEs), which specify the SID of a user or group of users, and whether access rights are to be allowed, denied, or audited. For resources on a print server, the ACEs specify operations including print, manage printers, and manage documents in a print queue.

The SD that is associated with the print server or print queue controls the creation of the context handle (PRINTER_HANDLE) and the outcome of operations that use the handle. These operations range from printing and job management to listening for notifications.

The SDs for a Windows-based print server are used to control the creation and deletion of print queues on the server; the installation of print system components, including printer drivers, print processors, and port monitors; and the enumeration of resources on the server. A server SD is not write-accessible to callers. In addition to being used to control callers' access to resources, the server SD is also used as the "parent" in the creation of the print queue's SD.

Note: The SD of the Windows-based print server is different from the SD that is applied on the named pipe of the spool. The SD of the spool's named pipe controls the RPC client's access to make RPC calls to the print server. The SD of the Windows-based print server is used to control the caller's permissions to perform various operations on the print server.

The SD of the print queue controls the setting of print queue properties, such as the port and printer driver that are used for printing; the rendering and device settings; or the sharing or security parameters. The printer SD allows auditing operations such as printing; managing printers and documents; reading and changing permissions; and taking ownership.

Each print job has an associated SD, which is created by using the SD of the print queue as parent. The user who submitted the document for printing is the owner of the print job and has permission to manage the print job during its lifetime.

When a PRINTER_HANDLE is opened for a specific printing resource, the caller must specify the access that is needed for subsequent operations, including printer or server administration; printing

on a printer or print server; and reading, writing, or administering a print job. If the caller has the specified permissions, the print handle is created and can be used for subsequent calls.

In addition to handle-based operations, the SD is used for access checks when enumerations, printer **driver package** installation, or other non-handle-based operations are performed.

7.1 Version-Specific Additions

The following sections list additions by each new operating system version, starting with the latest.

7.1.1 New in Windows Vista SP1 and Windows Server 2008

New Wsdmon actions in Windows Server 2008
"AssocDeviceMulticast"
"BackupPort"
"CheckCluster"
"DiscoverDevice"
"DriverAvailable"
"AssocDevice"
"AddPrinterPort"

7.1.2 New in Windows Vista and Windows Server 2008

[RPC DRIVER INFO 8](#)

[_DRIVER_INFO 8](#)

[JOB INFO 4](#)

[_JOB_INFO 4](#)

[RPC FORM INFO 2](#)

[_FORM_INFO 2](#)

[SPLCLIENT INFO 3](#)

[PRINTER STATUS SERVER OFFLINE](#)

[PORT DATA 2](#)

[PORT DATA LIST 1](#)

Support for IPv6 server names.

Tcpmon::XcvData actions
"SetDeviceIDOID"
"DeviceID"

Tcpmon::XcvData actions
"GetPortList"
"CleanupPort"

7.1.3 New in Windows XP SP1

[JOB ACCESS READ](#)

7.1.4 New in Windows XP and Windows Server 2003

[_DRIVER_INFO_7](#)

[RPC_BIDI_DATA](#)

[RPC_BIDI_REQUEST_CONTAINER](#)

[RPC_BIDI_REQUEST_DATA](#)

[RPC_BIDI_RESPONSE_CONTAINER](#)

[RPC_BIDI_RESPONSE_DATA](#)

[RPC_BINARY_CONTAINER](#)

[RpcSendRecvBidiData](#)

7.1.5 New in Windows 2000

[RPC_DRIVER_INFO_4](#)

[_DRIVER_INFO_4](#)

[_DRIVER_INFO_5](#)

[RPC_DRIVER_INFO_6](#)

[_DRIVER_INFO_6](#)

[PRINTER_ENUM_VALUES](#)

[PRINTER_INFO_7](#)

[_PRINTER_INFO_7](#)

[PRINTER_INFO_8](#)

[_PRINTER_INFO_8](#)

[PRINTER_INFO_9](#)

[_PRINTER_INFO_9](#)

[RpcAddPerMachineConnection](#)

[RpcAddPrinterDriverEx](#)

[RpcDeletePerMachineConnection](#)

[RpcDeletePrinterDataEx](#)

[RpcDeletePrinterDriverEx](#)

[RpcDeletePrinterKey](#)

[RpcEnumPerMachineConnections](#)

[RpcEnumPrinterDataEx](#)

[RpcEnumPrinterKey](#)

[RpcFlushPrinter](#)

[RpcGetPrinterDataEx](#)

[RpcSetPrinterDataEx](#)

[RpcXcvData](#)

[PRINTER_ATTRIBUTE_PUBLISHED](#)

[PRINTER_ENUM_HIDE](#)

[PRINTER_NOTIFY_FIELD_OBJECT_GUID](#)

7.1.6 New in Windows NT 4.0

[RPC_DRIVER_INFO_3](#)

[_DRIVER_INFO_3](#)

[JOB_INFO_3](#)

[_JOB_INFO_3](#)

[PORT_INFO_3](#)

[PRINTER_INFO_5](#)

[_PRINTER_INFO_5](#)

[PRINTER_INFO_6](#)

[_PRINTER_INFO_6](#)

[RpcAddPrinterEx](#)

[RpcDeletePrinterData](#)

[RpcEnumPrinterData](#)

[RpcOpenPrinterEx](#)

[RpcSetPort](#)

[SPLCLIENT_CONTAINER](#)

[SPLCLIENT_INFO_1](#)

[JOB_STATUS_RESTART](#)

[PRINTER_ATTRIBUTE_RAW_ONLY](#)

7.1.7 New in Windows NT 3.51

[RpcRemoteFindFirstPrinterChangeNotificationEx](#)

[RpcRouterRefreshPrinterChangeNotification](#)

[RpcRouterReplyPrinterEx](#)

7.1.8 New in Windows NT 3.5

[PORT_CONTAINER](#)

[PORT_INFO_FF](#)

[PORT_VAR_CONTAINER](#)

[PRINTER_INFO_4](#)

[_PRINTER_INFO_4](#)

[PRINTER_INFO_STRESS](#)

[_PRINTER_INFO_STRESS](#)

[RPC_V2_NOTIFY_INFO](#)

[RPC_V2_NOTIFY_INFO_DATA](#)

[RPC_V2_NOTIFY_INFO_DATA_DATA](#)

[RPC_V2_NOTIFY_OPTIONS](#)

[RPC_V2_NOTIFY_OPTIONS_TYPE](#)

[RPC_V2_UREPLY_PRINTER](#)

[RpcAddPortEx](#)

[RpcFindClosePrinterChangeNotification](#)

[RpcGetPrinterDriver2](#)

[RpcRemoteFindFirstPrinterChangeNotification](#)

[RpcReplyClosePrinter](#)

[RpcReplyOpenPrinter](#)

[RpcRouterRefreshPrinterChangeNotification](#)

[RpcRouterReplyPrinter](#)

[STRING_CONTAINER](#)

SYSTEMTIME_CONTAINER

PRINTER_CHANGE_FAILED_CONNECTION_PRINTER

JOB_NOTIFY_FIELD_BYTES_PRINTED

JOB_NOTIFY_FIELD_DATATYPE
JOB_NOTIFY_FIELD_DEVMODE
JOB_NOTIFY_FIELD_DOCUMENT
JOB_NOTIFY_FIELD_DRIVER_NAME
JOB_NOTIFY_FIELD_MACHINE_NAME
JOB_NOTIFY_FIELD_NOTIFY_NAME
JOB_NOTIFY_FIELD_PAGES_PRINTED
JOB_NOTIFY_FIELD_PARAMETERS
JOB_NOTIFY_FIELD_PORT_NAME
JOB_NOTIFY_FIELD_POSITION
JOB_NOTIFY_FIELD_PRINTER_NAME
JOB_NOTIFY_FIELD_PRINT_PROCESSOR
JOB_NOTIFY_FIELD_PRIORITY
JOB_NOTIFY_FIELD_SECURITY_DESCRIPTOR
JOB_NOTIFY_FIELD_START_TIME
JOB_NOTIFY_FIELD_STATUS
JOB_NOTIFY_FIELD_STATUS_STRING
JOB_NOTIFY_FIELD_SUBMITTED
JOB_NOTIFY_FIELD_TIME
JOB_NOTIFY_FIELD_TOTAL_BYTES
JOB_NOTIFY_FIELD_TOTAL_PAGES
JOB_NOTIFY_FIELD_UNTIL_TIME
JOB_NOTIFY_FIELD_USER_NAME

JOB_STATUS_BLOCKED_DEVO

JOB_STATUS_DELETED
JOB_STATUS_USER_INTERVENTION
PRINTER_ATTRIBUTE_DO_COMPLETE_FIRST

PRINTER_ATTRIBUTE_ENABLE_BIDI
PRINTER_ATTRIBUTE_ENABLE_DEVQ
PRINTER_ATTRIBUTE_KEEPPRINTEDJOBS
PRINTER_ATTRIBUTE_WORK_OFFLINE
PRINTER_CHANGE_SET_PRINTER_DRIVER
[PRINTER_NOTIFY_FIELD_ATTRIBUTES](#)
PRINTER_NOTIFY_FIELD_AVERAGE_PPM
PRINTER_NOTIFY_FIELD_BYTES_PRINTED
PRINTER_NOTIFY_FIELD_CJOBS
PRINTER_NOTIFY_FIELD_COMMENT
PRINTER_NOTIFY_FIELD_DATATYPE
PRINTER_NOTIFY_FIELD_DEFAULT_PRIORITY
PRINTER_NOTIFY_FIELD_DEVMODE
PRINTER_NOTIFY_FIELD_DRIVER_NAME
PRINTER_NOTIFY_FIELD_LOCATION
PRINTER_NOTIFY_FIELD_PAGES_PRINTED
PRINTER_NOTIFY_FIELD_PARAMETERS
PRINTER_NOTIFY_FIELD_PORT_NAME
PRINTER_NOTIFY_FIELD_PRINTER_NAME
PRINTER_NOTIFY_FIELD_PRINT_PROCESSOR
PRINTER_NOTIFY_FIELD_PRIORITY
PRINTER_NOTIFY_FIELD_SECURITY_DESCRIPTOR
PRINTER_NOTIFY_FIELD_SEPFILE
PRINTER_NOTIFY_FIELD_SERVER_NAME
PRINTER_NOTIFY_FIELD_SHARE_NAME
PRINTER_NOTIFY_FIELD_START_TIME
PRINTER_NOTIFY_FIELD_STATUS
PRINTER_NOTIFY_FIELD_TOTAL_BYTES
PRINTER_NOTIFY_FIELD_TOTAL_PAGES
PRINTER_NOTIFY_FIELD_UNTIL_TIME

PRINTER_NOTIFY_INFO_COLORMISMATCH

PRINTER_NOTIFY_INFO_DISCARDED

PRINTER_NOTIFY_INFO_DISCARDNOTED

PRINTER_NOTIFY_OPTIONS_REFRESH

PRINTER_STATUS_POWER_SAVE

PRINTER_STATUS_SERVER_UNKNOWN

[SPECIFIC_RIGHTS_ALL](#)

STANDARD_RIGHTS_ALL

SYNCHRONIZE

7.1.9 Irregular Version-Specific Behavior

The following API is only called over the wire by Windows NT 4.0 clients:

[RpcDeletePort](#)

8 Index

[DRIVER_FILE_INFO packet](#)
[DRIVER_INFO](#)
[DRIVER_INFO 1 packet](#)
[DRIVER_INFO 101 packet](#)
[DRIVER_INFO 2 packet](#)
[DRIVER_INFO 3 packet](#)
[DRIVER_INFO 4 packet](#)
[DRIVER_INFO 5 packet](#)
[DRIVER_INFO 6 packet](#)
[DRIVER_INFO 7 packet](#)
[DRIVER_INFO 8 packet](#)
[FORM_INFO](#)
[FORM_INFO 1 packet](#)
[FORM_INFO 2 packet](#)
[JOB_INFO](#)
[JOB_INFO 1 packet](#)
[JOB_INFO 2 packet](#)
[JOB_INFO 3 packet](#)
[JOB_INFO 4 packet](#)
[MONITOR_INFO](#)
[MONITOR_INFO 1 packet](#)
[MONITOR_INFO 2 packet](#)
[PORT_INFO](#)
[PORT_INFO 1 packet](#)
[PORT_INFO 2 packet](#)
[PRINTER_INFO](#)
[PRINTER_INFO 1 packet](#)
[PRINTER_INFO 2 packet](#)
[PRINTER_INFO 3 packet](#)
[PRINTER_INFO 4 packet](#)
[PRINTER_INFO 5 packet](#)
[PRINTER_INFO 6 packet](#)
[PRINTER_INFO 7 packet](#)
[PRINTER_INFO 8 packet](#)
[PRINTER_INFO 9 packet](#)
[PRINTER_INFO STRESS packet](#)

A

Abstract data model

[client](#)
[server](#)

[Adding a printer driver to a server](#)

[Adding a printer to a server](#)

[ADDJOB_INFO 1 packet](#)

All other bits SHOULD be zero. ([section 2.2.3.7](#), [section 2.2.3.7](#), [section 2.2.3.7](#))

[Applicability](#)

B

[BIDI_TYPE enumeration](#)

[Bidirectional communication data](#)

C

[Capability negotiation](#)

Client

[abstract data model](#)

[initialization](#)

[local events](#)

[message processing](#)

[overview](#)

[sequencing rules](#)

[timer events](#)

[timers](#)

[Client interaction with print server](#)

[Client-side notification-processing methods](#)

Command value table ([section 3.1.4.11.3.2](#), [section 3.1.4.11.4.2](#))

[Common IDL data types](#)

[CONFIG_INFO_DATA 1](#)

[Constants](#)

[Containers](#)

[Custom marshaled data types](#)

[Custom-Marshaled Data Types packet](#)

D

Data model - abstract

[client](#)
[server](#)

Data types

[common](#)
[custom marshaled](#)
[IDL](#)

[DATATYPES_INFO 1 packet](#)

[DELETE_PORT_DATA_1](#) ([section 3.1.4.11.3.1.1](#), [section 3.1.4.11.3.1.2](#))

[DEVMODE packet](#)

[DEVMODE_CONTAINER structure](#)

[Discovery methods](#)

[DOC_INFO 1 structure](#)

[DOC_INFO_CONTAINER structure](#)

[Document printing methods](#)

[DRIVER_CONTAINER structure](#)

[DRIVER_INFO](#)

[DRIVER_INFO 1 structure](#)

[DRIVER_INFO 2 structure](#)

E

[Enumerating jobs and modifying job settings](#)

[Enumerating printers](#)

[Examples](#)

F

[Form management methods](#)

[FORM_CONTAINER structure](#)

[FORM_INFO](#)

[FORM_INFO 1 structure](#)

[Full IDL](#)

G

[Generic Driver Extra Data](#)

[Glossary](#)

I

[IDL](#)

[IDL - data types](#)

[Implementer - security considerations](#)

[Index of security parameters](#)

[Informative references](#)

[Initialization](#)

[client](#)

[server](#)

[Introduction](#)

J

[Job management methods](#)

[Job scheduling](#)

[JOB_ACCESS_ADMINISTER](#)

[JOB_ACCESS_READ](#)

[JOB_CONTAINER structure](#)

[JOB_EXECUTE](#)

[JOB_INFO](#)

[JOB_INFO_1 structure](#)

[JOB_INFO_2 structure](#)

[JOB_INFO_3 structure](#)

[JOB_INFO_4 structure](#)

[JOB_NOTIFY_FIELD_BYTES_PRINTED](#)

[JOB_NOTIFY_FIELD_DATATYPE](#)

[JOB_NOTIFY_FIELD_DEVMODE](#)

[JOB_NOTIFY_FIELD_DOCUMENT](#)

[JOB_NOTIFY_FIELD_DRIVER_NAME](#)

[JOB_NOTIFY_FIELD_MACHINE_NAME](#)

[JOB_NOTIFY_FIELD_NOTIFY_NAME](#)

[JOB_NOTIFY_FIELD_PAGES_PRINTED](#)

[JOB_NOTIFY_FIELD_PARAMETERS](#)

[JOB_NOTIFY_FIELD_PORT_NAME](#)

[JOB_NOTIFY_FIELD_POSITION](#)

[JOB_NOTIFY_FIELD_PRINT_PROCESSOR](#)

[JOB_NOTIFY_FIELD_PRINTER_NAME](#)

[JOB_NOTIFY_FIELD_PRIORITY](#)

[JOB_NOTIFY_FIELD_SECURITY_DESCRIPTOR](#)

[JOB_NOTIFY_FIELD_START_TIME](#)

[JOB_NOTIFY_FIELD_STATUS](#)

[JOB_NOTIFY_FIELD_STATUS_STRING](#)

[JOB_NOTIFY_FIELD_SUBMITTED](#)

[JOB_NOTIFY_FIELD_TIME](#)

[JOB_NOTIFY_FIELD_TOTAL_BYTES](#)

[JOB_NOTIFY_FIELD_TOTAL_PAGES](#)

[JOB_NOTIFY_FIELD_UNTIL_TIME](#)

[JOB_NOTIFY_FIELD_USER_NAME](#)

[JOB_READ](#)

[JOB_STATUS_BLOCKED_DEVQ](#)

[JOB_STATUS_COMPLETE](#)

[JOB_STATUS_DELETED](#)

[JOB_STATUS_DELETING](#)

[JOB_STATUS_ERROR](#)

[JOB_STATUS_OFFLINE](#)

[JOB_STATUS_PAPEROUT](#)

[JOB_STATUS_PAUSED](#)

[JOB_STATUS_PRINTED](#)

[JOB_STATUS_PRINTING](#)

[JOB_STATUS_RESTART](#)

[JOB_STATUS_SPOOLING](#)

[JOB_STATUS_USER_INTERVENTION](#)

[JOB_WRITE](#)

L

[Local events](#)

[client](#)

[server](#)

M

[Managing printers](#)

[Members](#)

[rules](#)

[vendor-extensible](#)

[Members in custom-marshaled INFO structures](#)

[Members in INFO structures](#)

[Message processing](#)

[client](#)

[server](#)

[Messages](#)

[overview](#)

[transport](#)

[Monitor module methods](#)

[MONITOR_CONTAINER structure](#)

[MONITOR_INFO](#)

[MONITOR_INFO_1 structure](#)

[MONITOR_INFO_2 structure](#)

N

[Normative references](#)

[Notification methods](#)

O

[OEM Driver Extra Data](#)

[Overview \(synopsis\)](#)

P

[Parameters](#)

[Parameters - security index](#)

[Port management methods](#)

[Port monitor management methods](#)

[PORT_CONTAINER structure](#)

[PORT_DATA_2](#)

[PORT_DATA_LIST_1](#)

[PORT_INFO](#)

[PORT_INFO_1 structure](#)

[PORT_INFO_2 structure](#)

[PORT_INFO_3 structure](#)

[PORT_INFO_FF structure](#)

[PORT_VAR_CONTAINER structure](#)

[PostScript Driver Extra Data](#)

[Preconditions](#)

[Prerequisites](#)

[Print processor management methods](#)

[Print Ticket Driver Extra Data](#)

[Printer driver management methods](#)

[Printer management methods](#)

[Printer notification data](#)
[PRINTER_ACCESS_ADMINISTER](#)
[PRINTER_ACCESS_USE](#)
[PRINTER_ALL_ACCESS](#)
[PRINTER_ATTRIBUTE_DEFAULT](#)
[PRINTER_ATTRIBUTE_DIRECT](#)
[PRINTER_ATTRIBUTE_DO_COMPLETE_FIRST](#)
[PRINTER_ATTRIBUTE_ENABLE_BIDI](#)
[PRINTER_ATTRIBUTE_ENABLE_DEVQ](#)
[PRINTER_ATTRIBUTE_FAX](#)
[PRINTER_ATTRIBUTE_KEEPPRINTEDJOBS](#)
[PRINTER_ATTRIBUTE_LOCAL](#)
[PRINTER_ATTRIBUTE_NETWORK](#)
[PRINTER_ATTRIBUTE_PUBLISHED](#)
[PRINTER_ATTRIBUTE_QUEUED](#)
[PRINTER_ATTRIBUTE_RAW_ONLY](#)
[PRINTER_ATTRIBUTE_SHARED](#)
[PRINTER_ATTRIBUTE_TS](#)
[PRINTER_ATTRIBUTE_WORK_OFFLINE](#)
[PRINTER_CHANGE_ADD_FORM](#)
[PRINTER_CHANGE_ADD_JOB](#)
[PRINTER_CHANGE_ADD_PORT](#)
[PRINTER_CHANGE_ADD_PRINT_PROCESSOR](#)
[PRINTER_CHANGE_ADD_PRINTER](#)
[PRINTER_CHANGE_ADD_PRINTER_DRIVER](#)
[PRINTER_CHANGE_ALL](#)
[PRINTER_CHANGE_CONFIGURE_PORT](#)
[PRINTER_CHANGE_DELETE_FORM](#)
[PRINTER_CHANGE_DELETE_JOB](#)
[PRINTER_CHANGE_DELETE_PORT](#)
[PRINTER_CHANGE_DELETE_PRINT_PROCESSOR](#)
[PRINTER_CHANGE_DELETE_PRINTER](#)
[PRINTER_CHANGE_DELETE_PRINTER_DRIVER](#)
[PRINTER_CHANGE_FAILED_CONNECTION_PRINTER](#)
[PRINTER_CHANGE_FORM](#)
[PRINTER_CHANGE_JOB](#)
[PRINTER_CHANGE_PORT](#)
[PRINTER_CHANGE_PRINT_PROCESSOR](#)
[PRINTER_CHANGE_PRINTER](#)
[PRINTER_CHANGE_PRINTER_DRIVER](#)
[PRINTER_CHANGE_SET_FORM](#)
[PRINTER_CHANGE_SET_JOB](#)
[PRINTER_CHANGE_SET_PRINTER](#)
[PRINTER_CHANGE_SET_PRINTER_DRIVER](#)
[PRINTER_CHANGE_TIMEOUT](#)
[PRINTER_CHANGE_WRITE_JOB](#)
[PRINTER_CONTAINER structure](#)
[PRINTER_ENUM_CONNECTIONS](#)
[PRINTER_ENUM_CONTAINER](#)
[PRINTER_ENUM_DEFAULT](#)
[PRINTER_ENUM_EXPAND](#)
[PRINTER_ENUM_FAVORITE](#)
[PRINTER_ENUM_HIDE](#)
[PRINTER_ENUM_ICON1](#)
[PRINTER_ENUM_ICON2](#)
[PRINTER_ENUM_ICON3](#)
[PRINTER_ENUM_ICON4](#)
[PRINTER_ENUM_ICON5](#)
[PRINTER_ENUM_ICON6](#)
[PRINTER_ENUM_ICON7](#)
[PRINTER_ENUM_ICON8](#)
[PRINTER_ENUM_LOCAL](#)

[PRINTER_ENUM_NAME](#)
[PRINTER_ENUM_NETWORK](#)
[PRINTER_ENUM_REMOTE](#)
[PRINTER_ENUM_SHARED](#)
[PRINTER_ENUM_VALUES packet](#)
[PRINTER_EXECUTE](#)
[PRINTER_INFO](#)
[PRINTER_INFO_1 structure](#)
[PRINTER_INFO_2 structure](#)
[PRINTER_INFO_3 structure](#)
[PRINTER_INFO_4 structure](#)
[PRINTER_INFO_5 structure](#)
[PRINTER_INFO_6 structure](#)
[PRINTER_INFO_7 structure](#)
[PRINTER_INFO_8 structure](#)
[PRINTER_INFO_9 structure](#)
[PRINTER_INFO_STRESS structure](#)
[PRINTER_NOTIFY_FIELD_ATTRIBUTES](#)
[PRINTER_NOTIFY_FIELD_AVERAGE_PPM](#)
[PRINTER_NOTIFY_FIELD_BYTES_PRINTED](#)
[PRINTER_NOTIFY_FIELD_CJOBS](#)
[PRINTER_NOTIFY_FIELD_COMMENT](#)
[PRINTER_NOTIFY_FIELD_DATATYPE](#)
[PRINTER_NOTIFY_FIELD_DEFAULT_PRIORITY](#)
[PRINTER_NOTIFY_FIELD_DEVMODE](#)
[PRINTER_NOTIFY_FIELD_DRIVER_NAME](#)
[PRINTER_NOTIFY_FIELD_LOCATION](#)
[PRINTER_NOTIFY_FIELD_OBJECT_GUID](#)
[PRINTER_NOTIFY_FIELD_PAGES_PRINTED](#)
[PRINTER_NOTIFY_FIELD_PARAMETERS](#)
[PRINTER_NOTIFY_FIELD_PORT_NAME](#)
[PRINTER_NOTIFY_FIELD_PRINT_PROCESSOR](#)
[PRINTER_NOTIFY_FIELD_PRINTER_NAME](#)
[PRINTER_NOTIFY_FIELD_PRIORITY](#)
[PRINTER_NOTIFY_FIELD_SECURITY_DESCRIPTOR](#)
[PRINTER_NOTIFY_FIELD_SEPFILE](#)
[PRINTER_NOTIFY_FIELD_SERVER_NAME](#)
[PRINTER_NOTIFY_FIELD_SHARE_NAME](#)
[PRINTER_NOTIFY_FIELD_START_TIME](#)
[PRINTER_NOTIFY_FIELD_STATUS](#)
[PRINTER_NOTIFY_FIELD_TOTAL_BYTES](#)
[PRINTER_NOTIFY_FIELD_TOTAL_PAGES](#)
[PRINTER_NOTIFY_FIELD_UNTIL_TIME](#)
[PRINTER_NOTIFY_INFO_COLORMISMATCH](#)
[PRINTER_NOTIFY_INFO_DISCARDED](#)
[PRINTER_NOTIFY_INFO_DISCARDNOTED](#)
[PRINTER_NOTIFY_OPTIONS_REFRESH](#)
[PRINTER_READ](#)
[PRINTER_STATUS_BUSY](#)
[PRINTER_STATUS_DOOR_OPEN](#)
[PRINTER_STATUS_ERROR](#)
[PRINTER_STATUS_INITIALIZING](#)
[PRINTER_STATUS_IO_ACTIVE](#)
[PRINTER_STATUS_MANUAL_FEED](#)
[PRINTER_STATUS_NO_TONER](#)
[PRINTER_STATUS_NOT_AVAILABLE](#)
[PRINTER_STATUS_OFFLINE](#)
[PRINTER_STATUS_OUT_OF_MEMORY](#)
[PRINTER_STATUS_OUTPUT_BIN_FULL](#)
[PRINTER_STATUS_PAGE_PUNT](#)
[PRINTER_STATUS_PAPER_JAM](#)
[PRINTER_STATUS_PAPER_OUT](#)

[PRINTER STATUS PAPER PROBLEM](#)
[PRINTER STATUS PAUSED](#)
[PRINTER STATUS PENDING DELETION](#)
[PRINTER STATUS POWER SAVE](#)
[PRINTER STATUS PRINTING](#)
[PRINTER STATUS PROCESSING](#)
[PRINTER STATUS SERVER OFFLINE](#)
[PRINTER STATUS SERVER UNKNOWN](#)
[PRINTER STATUS TONER LOW](#)
[PRINTER STATUS USER INTERVENTION](#)
[PRINTER STATUS WAITING](#)
[PRINTER STATUS WARMING UP](#)
[PRINTER WRITE](#)
[Printing a job](#)
[PRINTPROCESSOR INFO 1 packet](#)

R

[Receiving notifications on printing events](#)
[RECTL structure](#)

References

[informative](#)
[normative](#)
[overview](#)
[REG_BINARY](#)
[REG_DWORD](#)
[REG_DWORD_BIG_ENDIAN](#)
[REG_DWORD_LITTLE_ENDIAN](#)
[REG_EXPAND_SZ](#)
[REG_LINK](#)
[REG_MULTI_SZ](#)
[REG_NONE](#)
[REG_QWORD](#)
[REG_QWORD_LITTLE_ENDIAN](#)
[REG_RESOURCE_LIST](#)
[REG_SZ](#)
[Relationship to other protocols](#)
[RPC_BIDI_DATA structure](#)
[RPC_BIDI_REQUEST_CONTAINER structure](#)
[RPC_BIDI_REQUEST_DATA structure](#)
[RPC_BIDI_RESPONSE_CONTAINER structure](#)
[RPC_BIDI_RESPONSE_DATA structure](#)
[RPC_BINARY_CONTAINER structure](#)
[RPC_DRIVER_INFO 3 structure](#)
[RPC_DRIVER_INFO 4 structure](#)
[RPC_DRIVER_INFO 6 structure](#)
[RPC_DRIVER_INFO 8 structure](#)
[RPC_FORM_INFO 2 structure](#)
[RPC_V2_NOTIFY_INFO structure](#)
[RPC_V2_NOTIFY_INFO_DATA structure](#)
[RPC_V2_NOTIFY_OPTIONS structure](#)
[RPC_V2_NOTIFY_OPTIONS_TYPE structure](#)
[RpcAbortPrinter method](#)
[RpcAddForm method](#)
[RpcAddJob method](#)
[RpcAddMonitor method](#)
[RpcAddPerMachineConnection method](#)
[RpcAddPortEx method](#)
[RpcAddPrinter method](#)
[RpcAddPrinterDriver method](#)
[RpcAddPrinterDriverEx method](#)

[RpcAddPrinterEx method](#)
[RpcAddPrintProcessor method](#)
[RpcClosePrinter method](#)
[RpcCreatePrinterIC method](#)
[RpcDeleteForm method](#)
[RpcDeleteMonitor method](#)
[RpcDeletePerMachineConnection method](#)
[RpcDeletePort method](#)
[RpcDeletePrinter method](#)
[RpcDeletePrinterData method](#)
[RpcDeletePrinterDataEx method](#)
[RpcDeletePrinterDriver method](#)
[RpcDeletePrinterDriverEx method](#)
[RpcDeletePrinterIC method](#)
[RpcDeletePrinterKey method](#)
[RpcDeletePrintProcessor method](#)
[RpcEndDocPrinter method](#)
[RpcEndPagePrinter method](#)
[RpcEnumForms method](#)
[RpcEnumJobs method](#)
[RpcEnumMonitors method](#)
[RpcEnumPerMachineConnections method](#)
[RpcEnumPorts method](#)
[RpcEnumPrinterData method](#)
[RpcEnumPrinterDataEx method](#)
[RpcEnumPrinterDrivers method](#)
[RpcEnumPrinterKey method](#)
[RpcEnumPrinters method](#)
[RpcEnumPrintProcessorDatatypes method](#)
[RpcEnumPrintProcessors method](#)
[RpcFindClosePrinterChangeNotification method](#)
[RpcFlushPrinter method](#)
[RpcGetForm method](#)
[RpcGetJob method](#)
[RpcGetPrinter method](#)
[RpcGetPrinterData method](#)
[RpcGetPrinterDataEx method](#)
[RpcGetPrinterDriver method](#)
[RpcGetPrinterDriver2 method](#)
[RpcGetPrinterDriverDirectory method](#)
[RpcGetPrintProcessorDirectory method](#)
[RpcOpenPrinter method](#)
[RpcOpenPrinterEx method](#)
[RpcPlayGdiScriptOnPrinterIC method](#)
[RpcReadPrinter method](#)
[RpcRemoteFindFirstPrinterChangeNotification method](#)
[RpcRemoteFindFirstPrinterChangeNotificationEx method](#)
[RpcReplyClosePrinter method](#)
[RpcReplyOpenPrinter method](#)
[RpcResetPrinter method](#)
[RpcRouterRefreshPrinterChangeNotification method](#)
[RpcRouterReplyPrinter method](#)
[RpcRouterReplyPrinterEx method](#)
[RpcScheduleJob method](#)
[RpcSendRecvBidiData method](#)
[RpcSetForm method](#)
[RpcSetJob method](#)
[RpcSetPort method](#)
[RpcSetPrinter method](#)
[RpcSetPrinterData method](#)
[RpcSetPrinterDataEx method](#)

[RpcStartDocPrinter method](#)
[RpcStartPagePrinter method](#)
[RpcWaitForPrinterChange method](#)
[RpcWritePrinter method](#)
[RpcXcvData method](#)

S

Security

[implementer considerations](#)
[overview](#)
[parameter index](#)

[SECURITY_CONTAINER structure](#)

Sequencing rules

[client](#)
[server](#)

Server

[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

[SERVER_ACCESS_ADMINISTER](#)

[SERVER_ACCESS_ENUMERATE](#)

[SERVER_ALL_ACCESS](#)

[SERVER_EXECUTE](#)

[SERVER_READ](#)

[SERVER_WRITE](#)

[Setting job information](#)

[SIZE structure](#)

[SPECIFIC_RIGHTS_ALL](#)

[SPLCLIENT_CONTAINER structure](#)

[SPLCLIENT_INFO](#)

[SPLCLIENT_INFO_1 structure](#)

[SPLCLIENT_INFO_3 structure](#)

[STANDARD_RIGHTS_ALL](#)

[STANDARD_RIGHTS_EXECUTE](#)

[STANDARD_RIGHTS_READ](#)

[STANDARD_RIGHTS_REQUIRED](#)

[STANDARD_RIGHTS_WRITE](#)

[Standards assignments](#)

[STRING_CONTAINER structure](#)

[Structures](#) ([section 3.1.4.11.3.1](#), [section 3.1.4.11.4.1](#))

[SYNCHRONIZE](#)

[SYSTEMTIME_CONTAINER structure](#)

T

[Tcpmon](#)

Timer events

[client](#)
[server](#)

Timers

[client](#)
[server](#)

[Transport - message](#)

U

[UNIVERSAL_FONT_ID structure](#)

V

[Vendor-extensible members](#)

[Versioning](#)

Version-specific additions

[irregular version-specific behavior](#)
[overview](#)

[Windows 2000](#)

[Windows NT 3.5](#)

[Windows NT 4.0](#)

[Windows Server 2008](#)

[Windows Vista and Windows Server 2008](#)

[Windows XP and Windows Server 2003](#)

[Windows XP SP1](#)

W

[Windows behaviors](#)

[WRITE_DAC](#)

[WRITE_OWNER](#)

[WSD_BACKUP_PORT_DATA](#)

[WSD_DRIVER_DATA](#)

[Wsdmon](#)