[MS-RAP]:
Remote Administration Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
12/18/2006	0.1		MCPP Milestone 2 Initial Availability
03/02/2007	1.0		MCPP Milestone 2
04/03/2007	1.1		Monthly release
05/11/2007	1.2		Monthly release
06/01/2007	1.2.1	Editorial	Revised and edited the technical content.

Date	Revision History	Revision Class	Comments
07/03/2007	1.3	Minor	Minor technical content changes.
07/20/2007	1.3.1	Editorial	Revised and edited the technical content.
08/10/2007	1.3.2	Editorial	Revised and edited the technical content.
09/28/2007	1.3.3	Editorial	Revised and edited the technical content.
10/23/2007	1.4	Minor	Made technical and editorial changes based on feedback.
11/30/2007	1.4.1	Editorial	Revised and edited the technical content.
01/25/2008	2.0	Major	Updated and revised the technical content.

Table of Contents

1 Introduction

This specification describes an extension of the Remote Administration Protocol. The protocol is included in the Windows operating system for compatibility reasons to perform remote administrative functions. The administrative functions include tasks such as **share** maintenance and printer maintenance on LAN Manager servers. In addition, the Common Internet File System (CIFS) Browser Protocol uses the Remote Administration Protocol to enumerate the servers on the network.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Little-Endian
NT LAN Manager Protocol (NTLM)
Print Queue
Share

The following terms are specific to this document:

Code Page: A table that describes a character set for a particular language. It maps logical character codes to single or multiple-byte character representations. It is used by an operating system to correctly display and print a language.

InfoLevel: A field in the Remote Administration Protocol command request that determines the structure that is returned in the response to the specified command.

Job ID: A 16-bit identifier used to identify a **print job** within a **print queue**.

Print Destinations: The list of drivers to which a **print queue** can print.

Print Job: The rendered page description language output data sent to a print device for a particular application request or user request.

Printer Separator Page: A page printed between separate **print jobs**.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as specified 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.

[CIFS] Leach, P. and Naik, D., "A Common Internet File System (CIFS/1.0) Protocol", March 1997, http://www.microsoft.com/about/legal/intellectualproperty/protocols/BSTD/CIFS/dr aft-leach-cifs-v1-spec-02.txt

If you have any trouble finding [CIFS], please check here.

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

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

[MS-SMB] Microsoft Corporation, "Server Message Block (SMB) 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

1.2.2 Informative References

[CIFSPRINT] Leach, P. and Naik, D., "CIFS Printing Specification Preliminary Draft", January 1997, http://hegel.ittc.ku.edu/topics/internet/internet-drafts/draft-l/draft-leach-cif s-print-spec-00.txt

[MS-BRWS] Microsoft Corporation, "Common Internet File System (CIFS) Browser Protocol Specification", July 2007.

[MS-NLMP] Microsoft Corporation, "NT LAN Manager (NTLM) Authentication Protocol Specification", June 2007.

[MS-RPRN] Microsoft Corporation, "Print System Remote Protocol Specification", June 2007.

[NTLM] Microsoft Corporation, "Microsoft NTLM", http://msdn2.microsoft.com/en-us/library/aa378749.aspx

If you have any trouble finding [NTLM], please check here.

[RAP] Leach, P. and Naik, D., "CIFS Remote Administration Protocol–Preliminary Draft", February 1997, ftp://ftp.microsoft.com/developr/drg/CIFS/cifsrap2.txt

[RYAN] Ryan, R. and Ryan, B., "LAN Manager: A Programmer's Guide, Version 2", Microsoft Press, July 1990, ISBN: 1556151667.

1.3 Protocol Overview (Synopsis)

The Remote Administration Protocol provides a simple remote procedure call (RPC)-like mechanism that enables clients to perform administrative functions on servers that implement the protocol. The Remote Administration Protocol allows the following:

- A client to retrieve an enumeration of the servers on the network.
- A server to provide an enumeration of the file shares that are available on the server.
- A server to return limited configuration information about the file and print services on the server.

A typical use might be a Windows 98 client retrieving a list of all shares that a server offers.

1.4 Relationship to Other Protocols

The Remote Administration Protocol is implemented using the Server Message Block (SMB) Protocol. The data flow for the Remote Administration Protocol is identical to the data flow for the SMB Protocol, as specified in [MS-SMB].

A subset of the Remote Administration Protocol is used by the CIFS Browser Protocol.

1.5 Prerequisites/Preconditions

The Remote Administration Protocol has the following preconditions:

- The SMB dialect negotiated between a Remote Administration Protocol client and a server MUST be for Microsoft LAN Manager version 1.0 or later, as specified in [MS-SMB], section 3.2.4.2.2.
- The Remote Administration Protocol also relies on a client establishing a connection to an SMB server. Before a client can issue Remote Administration Protocol commands, it MUST establish a connection to the server, and MUST successfully perform a TreeConnect SMB to the "IPC$" share on the server.

1.6 Applicability Statement

The Remote Administration Protocol is used when a client (designed to interoperate with Microsoft LAN Manager 1.0) needs to retrieve information on a server. If a server requires interoperability with such clients, it must implement this protocol.<1>

The Remote Administration Protocol is designed for 16-bit operating systems and, as such, is incapable of transmitting more than 64 KB of data in any protocol exchange.

1.7 Versioning and Capability Negotiation

This specification covers versioning issues in the following areas:

- Protocol Versions: The Remote Administration Protocol is supported in the following explicit dialects: LAN Manager 1.0, **NT LAN Manager (NTLM) Protocol** 0.12 (for more information, see [MS-NLMP]), and SMB. These dialects are specified in [MS-SMB] section 2.2, and the negotiation of such is specified in [MS-SMB] section 1.7. For more information on the NTLM Protocol, see [NTLM].
- Security and Authentication Methods: The Remote Administration Protocol uses the security and authentication methods already present in the SMB Protocol. The SMB Protocol supports the following authentication methods: LANMAN, NTLMv1, NTLMv2, and Kerberos.<2> These authentication methods are specified in [MS-SMB].
- Localization: The Remote Administration Protocol does not support localization or internationalization. Text strings are encoded in ASCII and are always transmitted as octets. If the octets are outside the ASCII range, 0x20-0x7F, the characters are interpreted in the **code page** of the processing system.
- Capability Negotiation: The Remote Administration Protocol has multiple modes that are implicitly detected by the Remote Administration Protocol at the command and protocol levels through mechanisms specified in [MS-SMB] section 2.2.

1.8 Vendor-Extensible Fields

There are no vendor-extensible fields in the Remote Administration Protocol. The commands that the Remote Administration Protocol processes may include vendor-extensible fields (version information, descriptive text, and so on).

This protocol uses Win32 error codes. These values are taken from the Windows error number space specified in [MS-ERREF]. Implementations should<3> reuse those values with their indicated meanings. Choosing any other value runs the risk of a collision in the future.

1.9 Standards Assignments

The Remote Administration Protocol utilizes a single parameter assignment: the *Name* parameter is assigned the case sensitive string "\PIPE\LANMAN". For more information, see section 3.

2 Messages

2.1 Transport

The Remote Administration Protocol is implemented using the SMB_COM_TRANSACTION functionality in the SMB Protocol. A client of the Remote Administration Protocol MUST first connect to the SMB server and exchange the SMB_COM_NEGOTIATE, SMB_COM_SESSION_SETUP_ANDX, and SMB_COM_TREE_CONNECT_ANDX commands to establish the connection, as specified in [MS-SMB] section 3.2.4.2.

2.2 Message Syntax

The Remote Administration Protocol is a request/response protocol. A Remote Administration Protocol request is carried in a single SMB_COM_TRANSACTION request, and the Remote Administration Protocol response is carried in the SMB_COM_TRANSACTION response that corresponds to the request, as specified in [CIFS] section 3.13.

All multiple-byte elements in the Remote Administration Protocol MUST be treated as **little-endian**, unless otherwise specified.

2.3 Information Levels

The Remote Administration Protocol supports the concept of an information level (or level of detail) required for a particular response. An information level is an unsigned 16-bit integer. A Remote Administration Protocol client requests a particular information level in a request, and the server responds with a structure in the **Data** field of the response corresponding to that information level. Numerically higher information levels provide more detailed information than lower information levels for a particular request/response pair. The following table specifies the requests, the supported information level for each request, and the response structures returned for this protocol.

Request	Information level	Response structure
NetServerEnum2	0x0000	NetServerInfo0
	0x0001	NetServerInfo1
NetServerEnum3	0x0000	NetServerInfo0
	0x0001	NetServerInfo1
NetShareEnum	0x0001	NetShareInfo
NetPrintQGetInfo	0x0002	PrintQueue2
	0x0003	PrintQueue3
NetPrintJobGetInfo	0x0001	PrintJobInfo0

2.4 String Field Length Limits

Many of the string elements specified in the Remote Administration Protocol have maximum length constraints associated with them. A client MUST NOT transmit strings that exceed the maximum

length, as specified in the following tables. A server MUST fail a request if it receives a string that exceeds the maximum length by returning the associated Remote Administration Protocol response message with the Win32 error code set to ERROR_INVALID_PARAMETER (0x0057).

All text strings are encoded in ASCII and are received and transmitted as sequences of octets. The following tables specify the maximum character length, in bytes, for the string elements (not including a null-terminator) in various Remote Administration Protocol commands and structures, if any are required for a particular string.

The following table lists the length limits for RAP commands.

RAP command	Field name	Maximum character length
NetServerEnum2, NetServerEnum3	ServerName	15
NetServerEnum2, NetServerEnum3	ServerComment	48
NetServerEnum2Request, NetServerEnum3Request	Domain	15
NetServerEnum3Request	FirstServerToReturn	15
NetShareInfo	NetworkName	12
NetPrintQGetInfoRequest	PrintQueueName	12
PrintQueue2, PrintQueue3	PrintQName	12
PrintQueue2, PrintQueue3	SeparatorPageFilename	48
PrintQueue2, PrintQueue3	PrintProcessorDllName	48
PrintQueue2, PrintQueue3	CommentString	48
PrintQueue2	PrinterDestinationsName	48
PrintJobInfo0, PrintJobInfo3	UserName	20
PrintJobInfo0, PrintJobInfo3	NotifyName	15
PrintJobInfo0, PrintJobInfo3	DataType	9
PrintJobInfo0, PrintJobInfo3	ParametersString	48
PrintJobInfo0, PrintJobInfo3	JobStatusString	48
PrintJobInfo0, PrintJobInfo3	JobComment	48

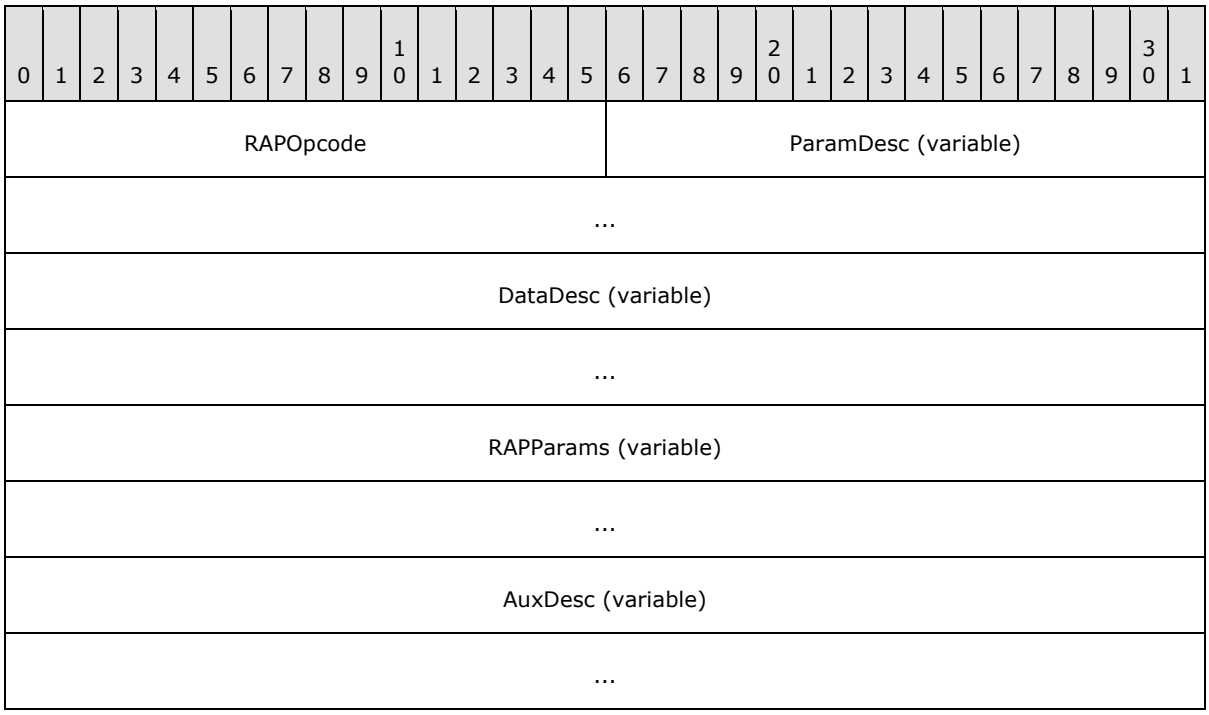
The following table lists the length limits for RAP structures.

RAP structure	Field name	Maximum characters
NetUserPasswordSet2Request	UserName	20
NetUserPasswordSet2Request	OldPassword	14
NetUserPasswordSet2Request	NewPassword	14

2.5 Message Definitions

2.5.1 RAP Request Message

Each Remote Administration Protocol request message MUST be transmitted in the parameters section of an SMB_COM_TRANSACTION protocol exchange (as specified in [CIFS] section 3.13). The Remote Administration Protocol request message MUST have the following format.



RAPOpcode (2 bytes): The operation code for the particular operation. For more information on valid operation codes, see 2.5.4.

ParamDesc (variable): This value MUST be a null-terminated ASCII descriptor string. The server SHOULD<4> validate that the **ParamDesc** value passed by the client matches what is specified by the **RAPOpcode**. For information on the origin of the descriptor string values, see section see section 4.2 of [RAP]. However, to implement this protocol, all that is required is to use the values specified herein.

DataDesc (variable): (Optional) If this value is specified, it MUST be a null-terminated ASCII descriptor string that describes the contents of the data returned to the client.<5> Certain **RAPOpcodes** specify a **DataDesc** field; for a list of Remote Administration Protocol commands that specify a **DataDesc** field, see section 2.5.5.

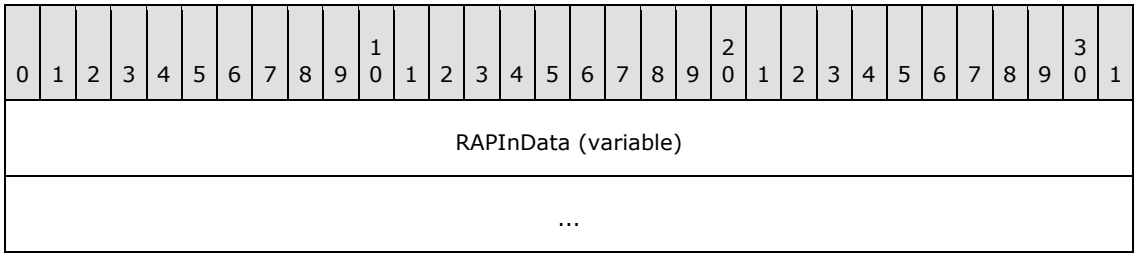
If no **DataDesc** field is specified for the Remote Administration Protocol command, this field MUST NOT be present. For the origin of the descriptor string values, see section 4.2.

RAPParams (variable): Remote Administration Protocol command-specific parameters, as specified in sections 2.5.5, 2.5.6, 2.5.7, 2.5.8, and 2.5.9.

AuxDesc (variable): (Optional) If this value is specified, it MUST be a null-terminated ASCII descriptor string that describes auxiliary data returned to the client.<6> If no **AuxDesc** field

is specified for the Remote Administration Protocol command, this field MUST NOT be present. For the origin of the descriptor string values, see section 4.2.

In addition, if the command specifies that it also uses the **Data** field of the SMB_COM_TRANSACTION, the format of the **Data** field MUST be:

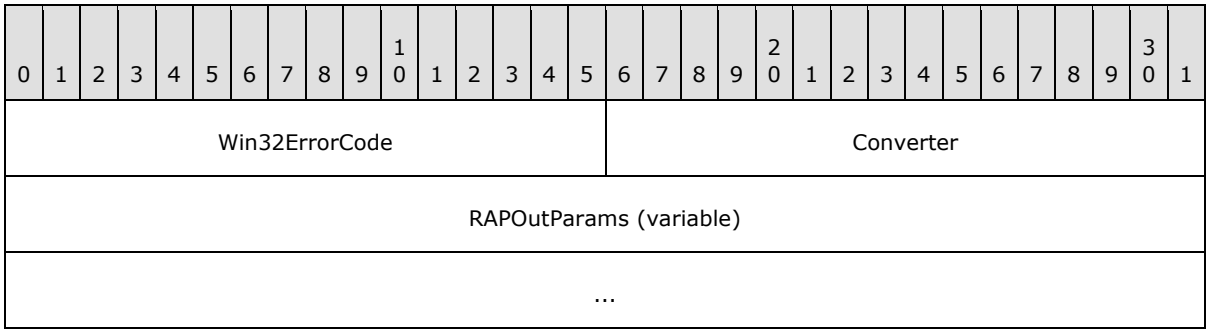


RAPInData (variable): Additional data for the Remote Administration Protocol request. This field MUST be present in the NetPrintJobSetInfoRequest command. This field MUST NOT be present in any other command.

2.5.2 RAP Response Message

The response to a Remote Administration Protocol command consists of two parts. The first is transmitted in the **Parameters** field of the SMB_COM_TRANSACTION response; the second is transmitted in the **Data** field of the same SMB_COM_TRANSACTION response (as specified in [CIFS] section 3.13).

The following MUST be the layout of the data in the SMB_COM_TRANSACTION response **Parameters** field.



Win32ErrorCode (2 bytes): This MUST be a 16-bit unsigned integer. It contains a Win32 error code representing the result of the Remote Administration Protocol command. The following table lists error codes that have particular meaning to the Remote Administration Protocol, as indicated in this specification.

Code/Value	Meaning
ERROR_SUCCESS 0x0000	No errors encountered.
ERROR_MORE_DATA 0x00EA	Additional data is available.
ERROR_INSUFFICIENT_BUFFER	The buffer supplied is too small.

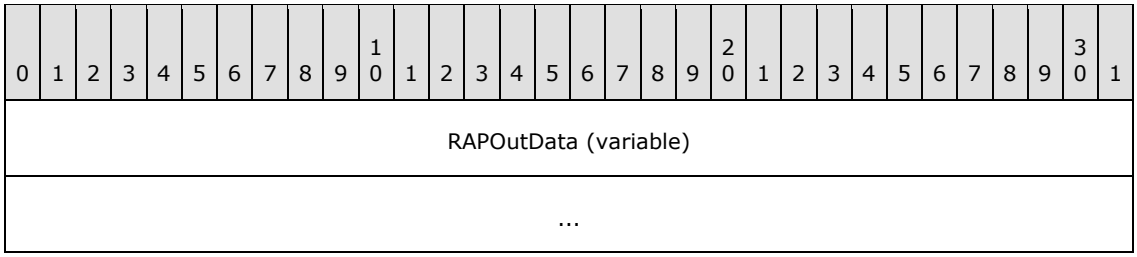
Code/Value	Meaning
0x007A	
ERROR_INVALID_LEVEL 0x007C	The specified information level is not supported.
ERROR_INVALID_PARAMETER 0x0057	Data from the client is invalid.

A Remote Administration Protocol server implementation MAY return Win32 error codes other than those listed in the preceding table. Any such error code SHOULD be drawn from the set of error codes specified in [MS-ERREF], and the client MUST treat any error code not explicitly listed in the preceding table as a failure.

Converter (2 bytes): This field MUST contain a 16-bit signed integer, which a client MUST subtract from the string offset contained in the low 16 bits of a variable-length field in the Remote Administration Protocol response buffer. This is to derive the actual byte offset from the start of the response buffer for that field.

RAPOutParams (variable): (Optional) If present, this structure MUST contain the response information for the Remote Administration Protocol command in the corresponding Remote Administration Protocol request message. Certain **RAPOpcodes** require a RAPOutParams structure; for Remote Administration Protocol commands that require a RAPOutParams structure, see sections 2.5.5, 2.5.6, 2.5.7, 2.5.8, and 2.5.9.

If the **Win32ErrorCode** in the **Parameters** field is either ERROR_SUCCESS (0x0000) or ERROR_MORE_DATA (0x00EA), the **Data** field of the SMB_COM_TRANSACTION MUST contain:



RAPOutData (variable): This is the response data for the Remote Administration Protocol operation. The contents of the RAPOutData structure varies according to the Remote Administration Protocol command and the parameters of each Remote Administration Protocol command. See Remote Administration Protocol responses for each Remote Administration Protocol command in sections 2.5.5, 2.5.6, 2.5.7, 2.5.8, and 2.5.9.

2.5.3 RAP Request/Response Summary Table

Some Remote Administration Protocol commands require the RAPOutParams structure, as specified in section 2.5.2. The following table specifies the message request and response for a specific Remote Administration Protocol command as well as the data structure for the response.

Command	Request	Response	Response data field
NetServerGetInfo	NetServerGetInfoRequest		
NetServerEnum2	NetServerEnum2Request	NetServerEnum2Response	NetServerInfo0, NetServerInfo1
NetServerEnum3	NetServerEnum3Request	NetServerEnum3Response	NetServerInfo0, NetServerInfo1
NetShareEnum	NetShareEnumRequest	NetShareEnumResponse	NetShareInfo
NetPrintQEnum	NetPrintQEnumRequest	NetPrintQEnumResponse	PrintQueue2, PrintJobInfo0
NetPrintQGetInfo	NetPrintQGetInfoRequest	NetPrintQGetInfoResponse	PrintQueue2, PrintJobInfo0, PrintQueue3, PrintJobInfo3
NetPrintJobGetInfo	NetPrintJobGetInfoRequest	NetPrintJobGetInfoResponse	PrintJobInfo0, PrintJobInfo3
NetPrintJobSetInfo	NetPrintJobSetInfoRequest		
NetPrintJobPause	NetPrintJobPauseRequest		
NetPrintJobContinue	NetPrintJobContinueRequest		
NetPrintJobDelete	NetPrintJobDeleteRequest		
NetUserPasswordSet2	NetUserPasswordSet2Request		
NetRemoteTOD	NetRemoteTODRequest	NetRemoteTODResponse	TimeOfDayInfo

2.5.4 RAP Opcodes

The following tables summarize Remote Administration Protocol command operation codes.

Server Commands

Command	Opcode
NetServerGetInfo	0x000D
NetServerEnum2	0x0068
NetServerEnum3	0x00D7

Share Commands

The NetShareEnum command has an opcode of 0x0000.

Print Commands

Command	Opcode
NetPrintQEnum	0x0045
NetPrintQGetInfo	0x0046
NetPrintJobSetInfo	0x0093
NetPrintJobGetInfo	0x004D
NetPrintJobPause	0x0052
NetPrintJobContinue	0x0053
NetPrintJobDelete	0x0051

User Commands

The NetUserPasswordSet2 command has an opcode of 0x0073.

Time Commands

The NetRemoteTOD command has an opcode of 0x005B.

2.5.5 RAP Server Commands

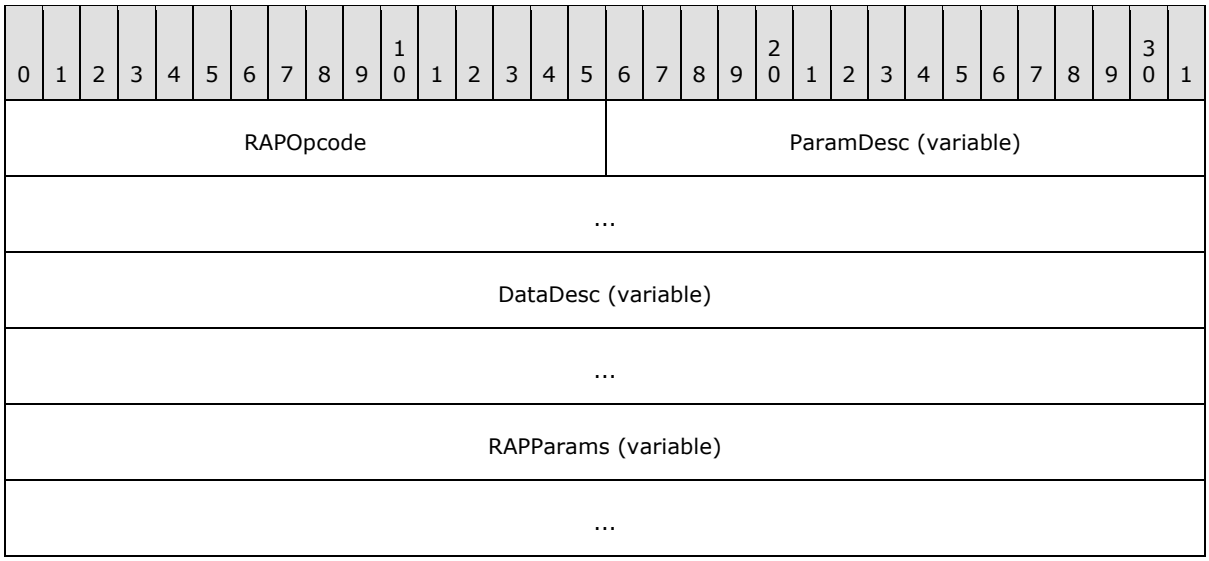
The following Remote Administration Protocol commands are for operations involving servers.

2.5.5.1 NetServerGetInfo Command

The NetServerGetInfo command returns information on the server.

2.5.5.1.1 RAP NetServerGetInfoRequest

The fields in the NetServerGetInfoRequest message MUST have the following format.

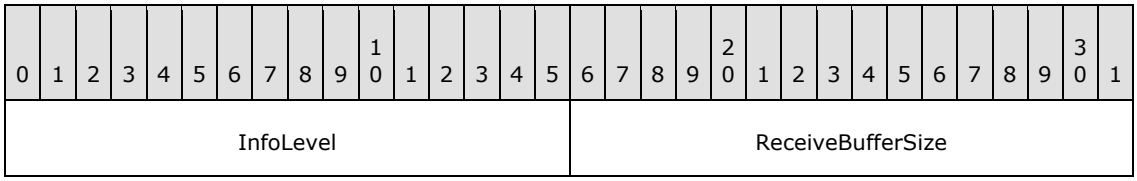


RAPOpcode (2 bytes): MUST be set to 0x000D. For more information, see section 2.5.1.

ParamDesc (variable): MUST be set to "WrLh". For more information, see section 2.5.1.

DataDesc (variable): MUST be set to "B16BBDz". For more information, see section 2.5.1.

RAPParams (variable): The RAPParams structure MUST have the following format.

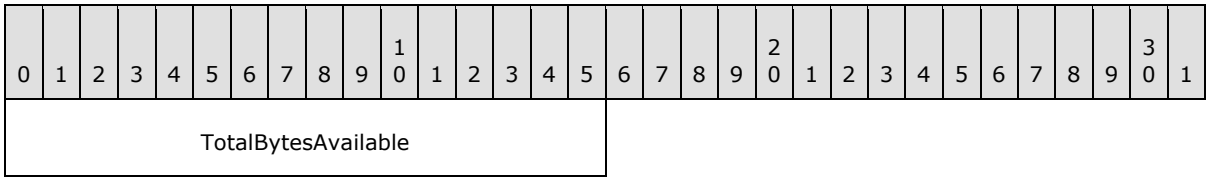


InfoLevel (2 bytes): A 16-bit unsigned integer that MUST specify the information level for the NetServerGetInfoRequest.

ReceiveBufferSize (2 bytes): A 16-bit unsigned integer that MUST represent the maximum number of bytes of data that may be returned in the **Data** field of the SMB_COM_TRANSACTION response to the command.

2.5.5.1.2 RAP NetServerGetInfoResponse

The **RAPOutParams** RAP response to the NetServerGetInfo command is as follows.



TotalBytesAvailable (2 bytes): A 16-bit unsigned integer that MUST represent the number of bytes required to hold the server information requested.

If the InfoLevel specified in the NetServerGetInfo is 0, and the response is not an error, the RAPOutData field of the SMB_COM_TRANSACTION response MUST be filled with a NetServerInfo0 structure.

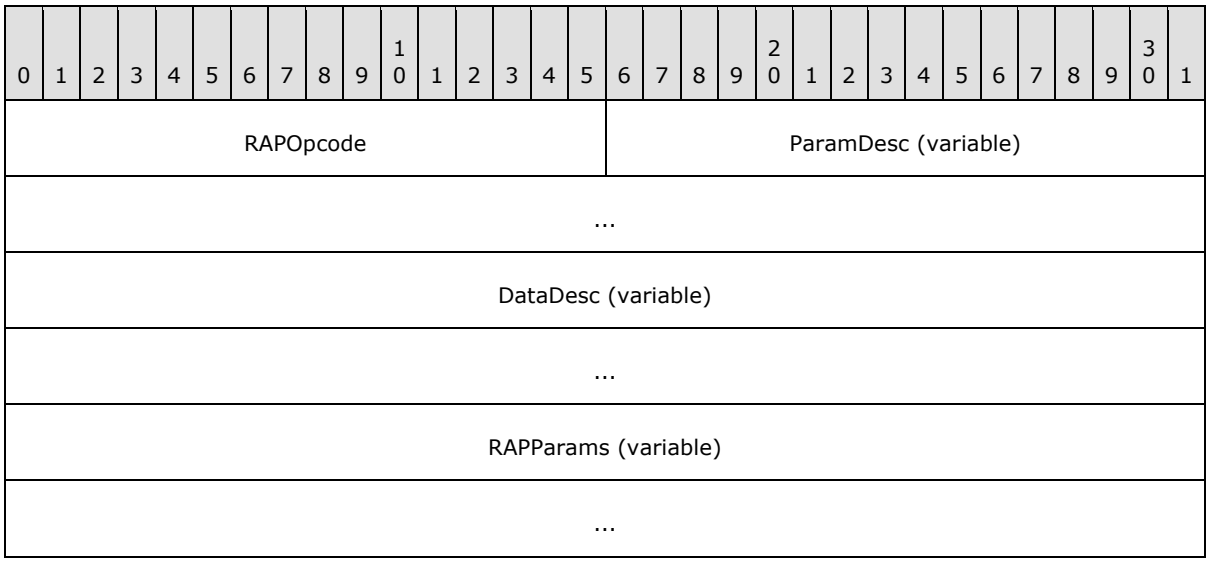
If the InfoLevel specified in the NetServerGetInfo is 1, and the response is not an error, the RAPOutData field of the SMB_COM_TRANSACTION response MUST be filled with a NetServerInfo1 structure.

2.5.5.2 NetServerEnum2

The NetServerEnum2 command specifies that the server is to return its list of servers to the client.

2.5.5.2.1 RAP NetServerEnum2Request

The fields in the NetServerEnum2Request message MUST be set as follows:

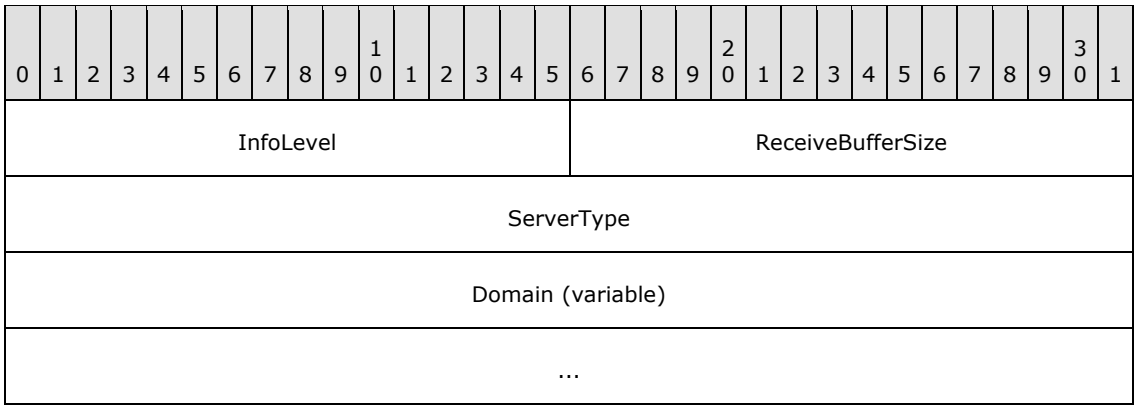


RAPOpcode (2 bytes): MUST be set to 0x0068. For more information, see section 2.5.1.

ParamDesc (variable): MUST be set to "WrLehDz", or "WrLehDO" if the Domain parameter is not specified. . For more information, see section 2.5.1.

DataDesc (variable): If **InfoLevel** (below) is set to 0x0000, this MUST be set to "B16"; if InfoLevel is set to 0x0001, this MUST be set to "B16BBDz". For more information, see section 2.5.1.

RAPParams (variable): The RAPParams structure MUST be as follows:

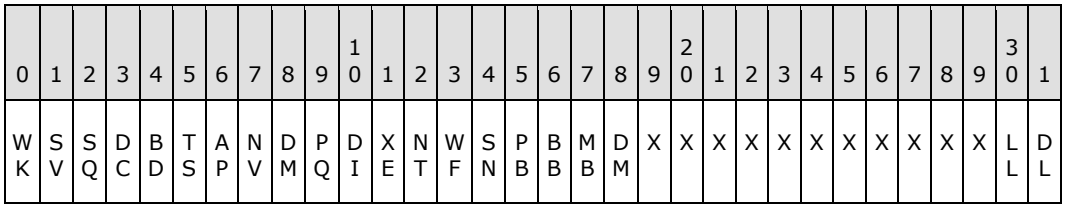


InfoLevel (2 bytes): A 16-bit unsigned integer that MUST specify the information level for the NetServerEnum2Request.

ReceiveBufferSize (2 bytes): A 16-bit unsigned integer that MUST represent the maximum number of bytes of data that may be returned in the **Data** field of the SMB_COM_TRANSACTION response to the command.

ServerType (4 bytes): A 32-bit set of flags used to filter servers in the response to the NetServerEnum2 command. The **ServerType** field MUST be a bitmask composed of the following possible values:

All bits labeled X SHOULD be set to 0 when sent and MUST be ignored when received.



Where the bits are defined as:

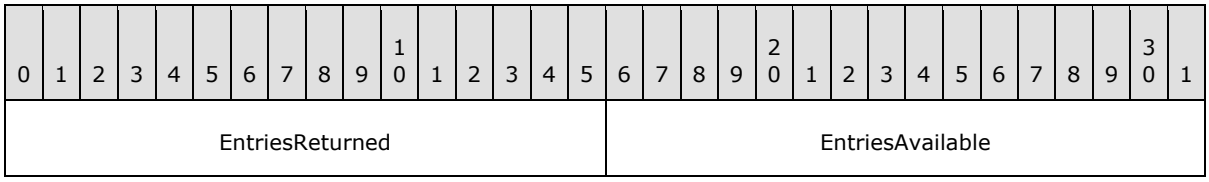
Value	Description
WK	The server is acting as a CIFS client.
SV	The server is acting as a CIFS server.
SQ	The server is functioning as a SQL server.
DC	The server is running as a domain controller.
BD	The server is running as a backup domain controller.
TS	The server is running a time-source service.
AP	The server is running the Apple File Protocol.
NV	The server is running the NetWare File Protocol.
DM	The server is a member of a domain.

Value	Description
PQ	The server is sharing printer queues.
DI	The server is running remote dial-in services.
XE	The server is running on the Xenix operating system.
NT	The server is running a version of Windows NT.
WF	The server is running a version of the Windows for Workgroups operating system.
SN	The server is running a version of Windows NT Advanced Server.
PB	The server is a potential browser server. For more information on the potential browser server role, see [MS-BRWS].
BB	The server is a backup browser server. For more information on the backup browser server role, see [MS-BRWS].
MB	The server is a master browser server. For more information on the master browser server role, see [MS-BRWS].
DM	The server is a domain master browser server. For more information on the domain master browser role server, see [MS-BRWS].
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
LL	If this flag is present in the NetServerEnum2Request, the server MUST return only computers that are on the same subnet as the current server.
DL	If this flag is present in the NetServerEnum2Request, the server MUST return the list of domains on the network. If this flag is present, all the flags other than the LL flag MUST be 0.

Domain (variable): If the ParamDesc is "WrLehDz", this field must contain a null-terminated ASCII string that MUST represent the name of the workgroup or domain for which to enumerate computers. If the ParamDesc is "WrLehD0", than this field MUST not be present. If this string is not present or is empty (a single null byte), the server MUST return the list of servers for the server's current domain or workgroup.

2.5.5.2.2 RAP NetServerEnum2Response

The RAPParams structure for the NetServerEnum2Response command MUST be as follows:



EntriesReturned (2 bytes): A 16-bit unsigned integer that MUST represent the number of fixed-size NetServerInfo0 or NetServerInfo1 data structures returned in the **Data** field of the SMB_COM_TRANSACTION response to the NetShareEnumRequest.

EntriesAvailable (2 bytes): A 16-bit unsigned integer that MUST represent the number of servers available on the server.

For error conditions and error responses, see section 3.2.5.11.

If the **InfoLevel** specified in the NetServerEnum2Request is 0x0000, the **RAPOutData** field of the SMB_COM_TRANSACTION response MUST be filled with an array of EntriesReturned NetServerInfo0 structures.

If the InfoLevel specified in the NetServerEnum2Request is 0x0001, the **RAPOutData** field of the SMB_COM_TRANSACTION response MUST be filled with an array of EntriesReturned NetServerInfo1 structures.

2.5.5.3 NetServerEnum3 Command

The NetServerEnum3 command specifies that the server MUST return to the client a list of servers that exist on the network.

2.5.5.3.1 RAP NetServerEnum3Request

The fields in the NetServerEnum3Request message MUST be set as follows:

RAPOpcode: MUST be set to 0x00D7.

ParamDesc: MUST be set to "WrLehDzz".

DataDesc: If the **InfoLevel** (below) is set to 0x0000, this MUST be set to "B16"; if the **InfoLevel** is set to 0x0001, this MUST be set to "B16BBDz".

RAPParams: The RAPParams structure MUST be 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
InfoLevel																ReceiveBufferSize															
ServerType																															
FirstNameToReturn (variable)																															
...																															
Domain (variable)																															
...																															

InfoLevel (2 bytes): A 16-bit unsigned integer that MUST specify the information level for the NetServerEnum3Request.

ReceiveBufferSize (2 bytes): A 16-bit unsigned integer that MUST represent the maximum bytes of data that may be returned in the **Data** field of the SMB_COM_TRANSACTION response to the command.

ServerType (4 bytes): A 32-bit set of flags used to filter servers in the response to the NetServerEnum2 command. The **ServerType** field MUST be a bitmask composed of the following possible values:

All bits labeled X SHOULD be set to 0 when sent, and MUST be ignored when received.

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
WK	SV	SQ	DC	BD	TS	AP	NV	DM	PQ	DI	XE	NT	WF	SN	PB	BB	MB	DM	X	X	X	X	X	X	X	X	X	X	X	LL	DL

Where the bits are defined as:

Value	Description
WK	The server is acting as a CIFS client.
SV	The server is acting as a CIFS server.
SQ	The server is functioning as a SQL server.
DC	The server is running as a domain controller.
BD	The server is running as a backup domain controller.
TS	The server is running a time-source service.

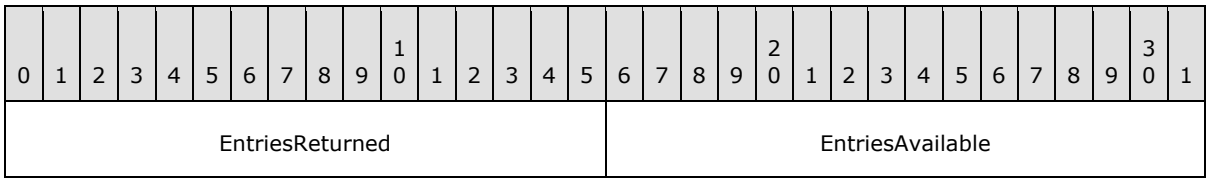
Value	Description
AP	The server is running the Apple File Protocol.
NV	The server is running the NetWare File Protocol.
DM	The server is a member of a domain.
PQ	The server is sharing printer queues.
DI	The server is running remote dial-in services.
XE	The server is running on the Xenix operating system.
NT	The server is running a version of Windows NT.
WF	The server is running a version of the Windows for Workgroups operating system.
SN	The server is running a version of Windows 2000 Advanced Server.
PB	The server is a potential browser server. For more information on the potential browser server role, see [MS-BRWS].
BB	The server is a backup browser server. For more information on the backup browser server role, see [MS-BRWS].
MB	The server is a master browser server. For more information on the master browser server role, see [MS-BRWS].
DM	The server is a domain master browser server. For more information on the domain master browser role server, see [MS-BRWS].
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
X	Unused.
LL	If this flag is present in the NetServerEnum3Request, the server MUST return only computers that are on the same subnet as the current server.
DL	If this flag is present in the NetServerEnum3Request, the server MUST return the list of domains on the network, and all flags other than the LL flag MUST be 0.

FirstNameToReturn (variable): This field MUST contain a null-terminated ASCII string with a maximum length of 16 bytes, including the null-terminator. This string MUST specify the name of the first server that the RAP server MUST return in its enumeration. If this parameter is empty (a single null byte), the server MUST return entries starting with the first server in the list. See section 3.2.5.13.

Domain (variable): A null-terminated ASCII string that MUST represent the name of the workgroup or domain for which to enumerate computers.

2.5.5.3.2 RAP NetServerEnum3Response

The RAPOutParams structure for the NetServerEnum3Response command MUST be as follows:



EntriesReturned (2 bytes): A 16-bit unsigned integer that MUST represent the number of fixed-size NetServerInfo0 or NetServerInfo1 data structures returned in the **Data** field of the SMB_COM_TRANSACTION response to the NetServerEnum3Request.

EntriesAvailable (2 bytes): A 16-bit unsigned integer that MUST represent the total number of servers available for enumeration on this network.

For error conditions and error responses, see section 3.2.5.13.

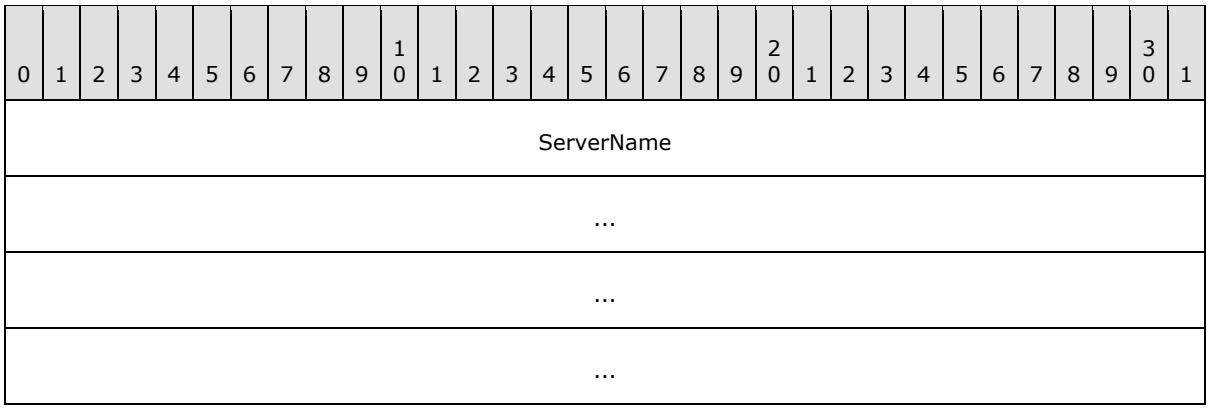
If the InfoLevel specified in the NetServerEnum3Request is 0x0000, the **RAPOutData** field of the SMB_COM_TRANSACTION response MUST be filled with an array of **EntriesReturned** NetServerInfo0 structures.

If the InfoLevel specified in the NetServerEnum3Request is 0x0001, the **RAPOutData** field of the SMB_COM_TRANSACTION response MUST be filled with an array of **EntriesReturned** NetServerInfo1 structures.

2.5.5.4 RAP Server Response Structures

2.5.5.4.1 RAP NetServerInfo0 Data Structure

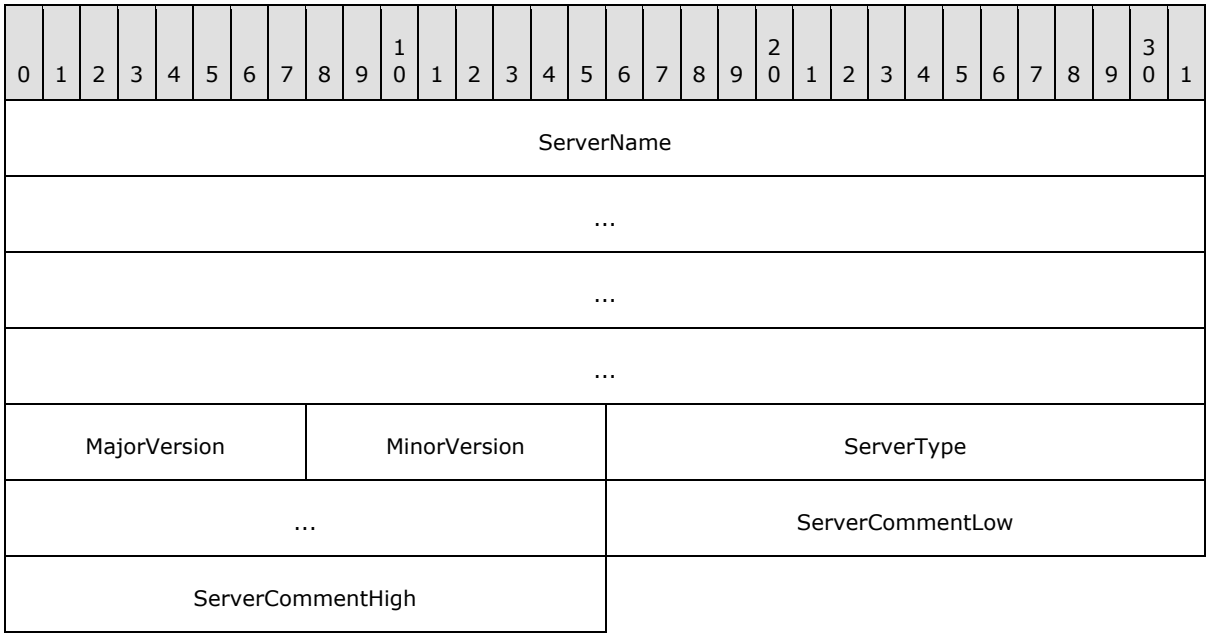
The NetServerInfo0 structure MUST be returned by the server in the **Data** field of the SMB_COM_TRANSACTION response that corresponds to a NetServerEnum2 command or a NetServerEnum3 command when the *InfoLevel* parameter to the command is 0x0000.



ServerName (16 bytes): A 16-character null-terminated ASCII string that MUST contain the NetBIOS name (as specified in [RFC1001] section 5.2) of the server. The **ServerName** field MUST be padded to 16 bytes with null characters.

2.5.5.4.2 RAP NetServerInfo1 Data Structure

The NetServerInfo1 structure is returned by the server in the **Data** field of the SMB_COM_TRANSACTION response that corresponds to a NetServerEnum2 command or a NetServerEnum3 command when the *InfoLevel* parameter to the command is 0x0001.



ServerName (16 bytes): A 16-character null-terminated ASCII string that MUST contain the NetBIOS name of the server (as specified in [RFC1001] section 5.2). The **ServerName** field MUST be padded to 16 bytes with null characters.

MajorVersion (1 byte): An 8-bit unsigned integer that MUST represent the major version of the specified server.<7>

MinorVersion (1 byte): An 8-bit unsigned integer that MUST represent the minor version of the specified server.<8>

ServerType (4 bytes): A 32-bit unsigned integer that MUST specify the type of software the computer is running. This field has the same syntax and semantics as the **ServerType** specified in section 2.5.5.2.1.

ServerCommentLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string allocated in the response block (see section 2.5.10) that MUST specify the purpose of the server.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field specified in section 2.5.2 from the **ServerCommentLow** value, and then use that result as the offset within the response.

ServerCommentHigh (2 bytes): Unused. Set to any arbitrary value on send and MUST be ignored on receipt.

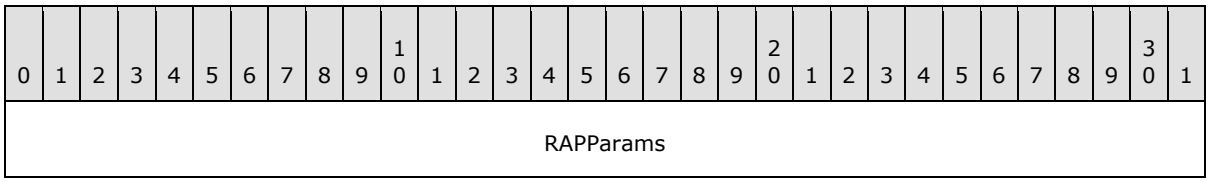
2.5.6 RAP Share Commands

2.5.6.1 NetShareEnum Command

The NetShareEnum command MUST return to the client information on each list of shared resources.

2.5.6.1.1 RAP NetShareEnumRequest

The fields in the NetShareEnumRequest message MUST be set as follows:

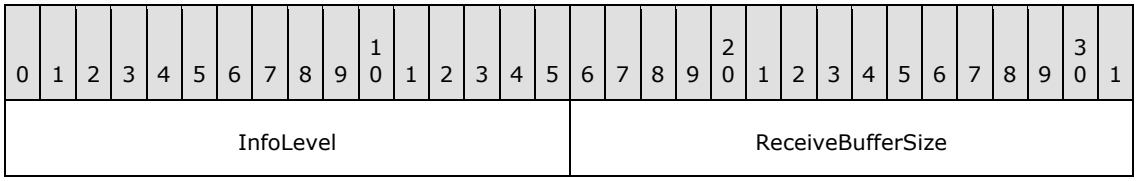


RAPOpcode: MUST be set to 0x0000.

ParamDesc: MUST be set to "WrLeh".

DataDesc: MUST be set to "B13".

RAPParams (4 bytes): The RAPParams structure MUST be as follows:

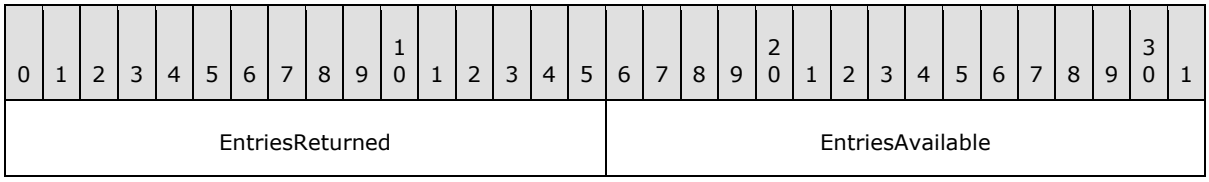


InfoLevel (2 bytes): A 16-bit unsigned integer that MUST specify the information level for NetShareEnumRequest.

ReceiveBufferSize (2 bytes): A 16-bit unsigned integer that MUST represent the maximum number of bytes of data that may be returned in the **Data** field of the SMB_COM_TRANSACTION response to the command.

2.5.6.1.2 RAP NetShareEnumResponse

The RAPOutParams structure for the NetShareEnum command MUST be as follows:



EntriesReturned (2 bytes): A 16-bit unsigned integer that MUST represent the number of fixed size NetShareInfo0, NetShareInfo1, or NetShareInfo2 data structures returned in the **Data** field of the SMB_COM_TRANSACTION response to the Remote Administration Protocol NetShareEnumRequest.

EntriesAvailable (2 bytes): A 16-bit unsigned integer that MUST represent the number of shares on the server.

For error conditions and error responses, see section 3.2.5.1.

If the **InfoLevel** specified in the NetShareEnumRequest is 0, the **RAPOutData** field of the SMB_COM_TRANSACTION response MUST be filled with an array of **EntriesReturned** NetShareInfo0 structures.

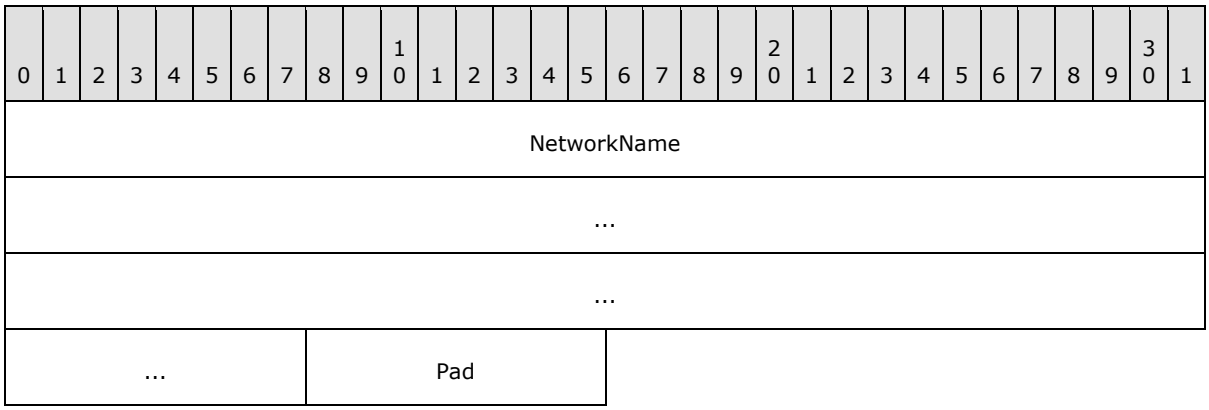
If the **InfoLevel** specified in the NetShareEnumRequest is 1, the **RAPOutData** field of the SMB_COM_TRANSACTION response MUST be filled with an array of **EntriesReturned** NetShareInfo1 structures.

If the **InfoLevel** specified in the NetShareEnumRequest is 2, the **RAPOutData** field of the SMB_COM_TRANSACTION response MUST be filled with an array of **EntriesReturned** NetShareInfo2 structures.

2.5.6.2 RAP Share Response Structures

2.5.6.2.1 NetShareInfo0

The NetShareInfo0 data structure has the following fields:



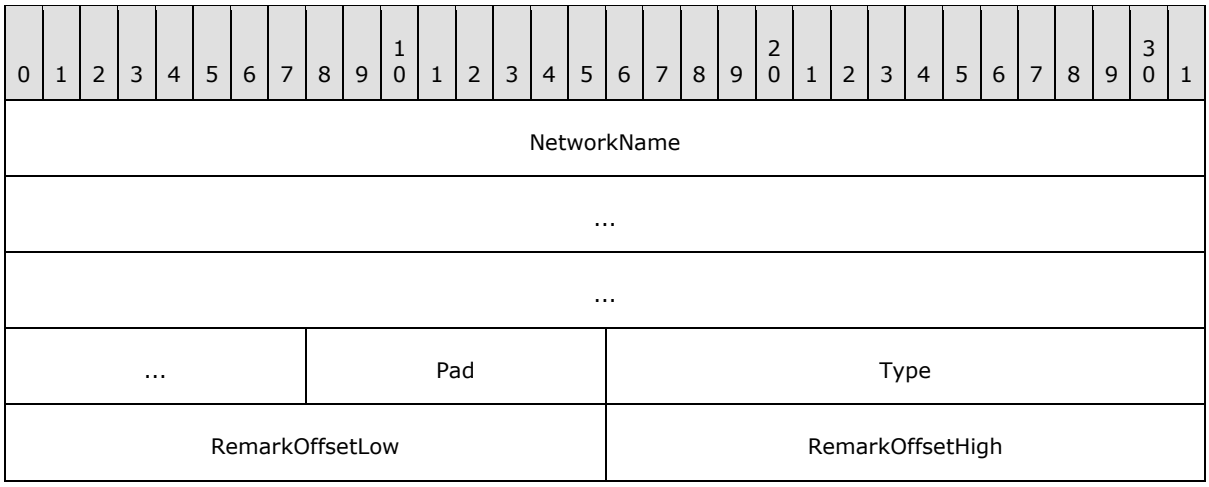
NetworkName (13 bytes): A 13-character null-terminated ASCII string that MUST specify the name of the share. If the name is shorter than 13 bytes, the NetworkName field MUST be

filled with null characters up to 13 bytes in length. If the name of the share is longer than 13 bytes, it MUST NOT be included in the enumeration.

Pad (1 byte): SHOULD be zero on send, and MUST be ignored on receipt.

2.5.6.2.2 NetShareInfo1

The NetShareInfo1 data structure has the following fields:



NetworkName (13 bytes): A 13-character, null-terminated ASCII string that MUST specify the name of the share. If the name is shorter than 13 bytes, the **NetworkName** field MUST be filled with null characters up to 13 bytes in length. If the name of the share is longer than 13 bytes, it MUST NOT be included in the enumeration.

Pad (1 byte): This field SHOULD be 0 on send and MUST be ignored on receipt.

Type (2 bytes): A 16-bit unsigned integer that MUST specify the type of the share. The **Type** field has the following possible values:

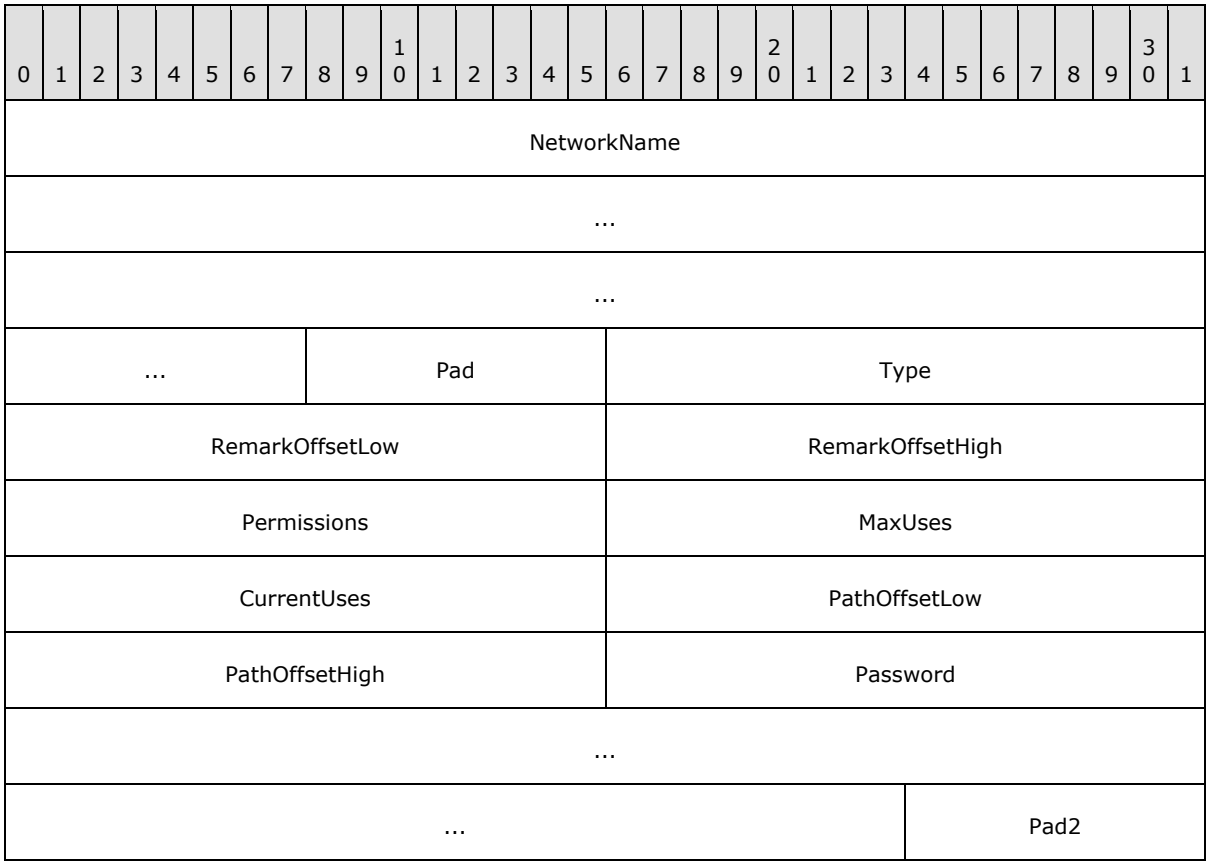
Value	Meaning
0x0000	Disk directory tree
0x0001	Printer queue
0x0002	Communications device
0x0003	Interprocess communication (IPC)

RemarkOffsetLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string allocated in the response block (see section 2.5.10) that MUST specify the purpose of the share. Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2 from the **RemarkOffsetLow** value, and then use that result as the offset within the response.

RemarkOffsetHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

2.5.6.2.3 NetShareInfo2

The NetShareInfo2 data structure has the following fields:



NetworkName (13 bytes): A 13-character null-terminated ASCII string that MUST specify the name of the share. If the name is shorter than 13 bytes, the NetworkName field MUST be filled with null characters up to 13 bytes in length. If the name of the share is longer than 13 bytes, it MUST NOT be included in the enumeration.

Pad (1 byte): SHOULD be zero on send, and MUST be ignored on receipt.

Type (2 bytes): A 16-bit unsigned integer that MUST specify the type of the share. The possible values for Type are:

Value	Meaning
0x0000	Disk Directory Tree
0x0001	Printer Queue
0x0002	Communications Device
0x0003	Inter-Process Communication (IPC)

RemarkOffsetLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset in bytes from the start of the response to a null-terminated ASCII string allocated in the response block (see section 2.5.10) that MUST specify the purpose of the share. Before using this value, the RAP client MUST subtract the Converter field specified in section 2.5.2 from the RemarkOffsetLow value, and then use that result as the offset within the response.

RemarkOffsetHigh (2 bytes): Unused. Set to an arbitrary value on send, and MUST be ignored on receipt.

Permissions (2 bytes): Obsolete value representing the access allowed in share-level security scenarios. This value SHOULD be set to 0 on send, and MUST be ignored on receipt.

MaxUses (2 bytes): The maximum number of users that are allowed to concurrently access this share.

CurrentUses (2 bytes): The current number of users accessing this share.

PathOffsetLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset in bytes from the start of the response to a null-terminated ASCII string allocated in the response block (see section 2.5.10) that MUST specify the local path of the share on the server. Before using this value, the RAP client MUST subtract the Converter field specified in section 2.5.2 from the PathOffsetLow value, and then use that result as the offset within the response.

PathOffsetHigh (2 bytes): Unused. Set to an arbitrary value on send, and MUST be ignored on receipt.

Password (9 bytes): A null-terminated ASCII string containing the password for password-protected shares. This value is only used for legacy share-level security, and SHOULD be set to an empty string.

Pad2 (1 byte): SHOULD be zero on send, and MUST be ignored on receipt.

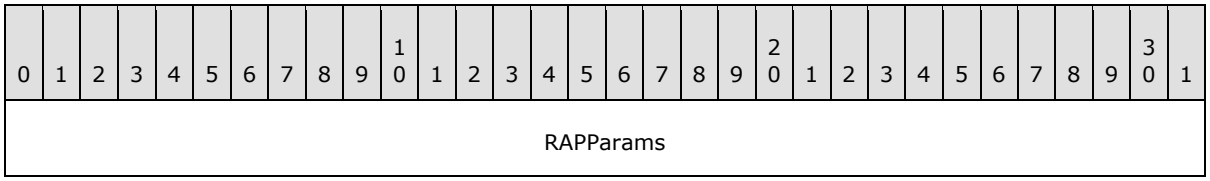
2.5.7 RAP Print Commands

2.5.7.1 NetPrintQEnum Command

The NetPrintQEnum command enables the server to return information that is an enumeration of the **print queues** on the server.

2.5.7.1.1 RAP NetPrintQEnumRequest

The fields in the NetPrintQEnumRequest message MUST be set as follows:



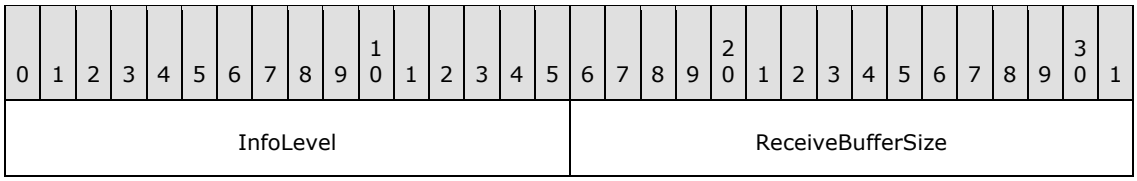
RAPOpcode: MUST be set to 0x0045.

ParamDesc: MUST be set to "WrLeh".

DataDesc: MUST be set to "B13BWWWzzzzzWN".

AuxDesc: MUST be set to "WB21BB16B10zWWzDDz".

RAPParams (4 bytes): The RAPParams structure MUST be as follows:

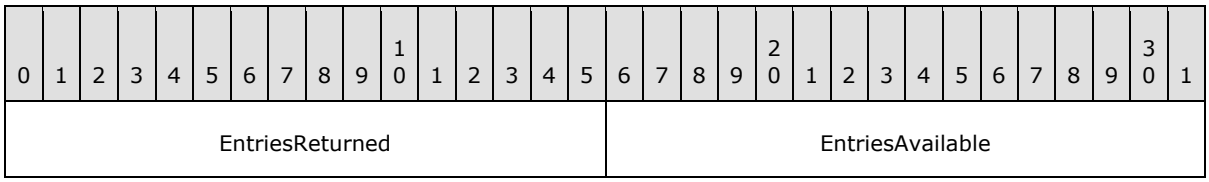


InfoLevel (2 bytes): A 16-bit unsigned integer that MUST specify the information level for NetPrintQEnumRequest.

ReceiveBufferSize (2 bytes): A 16-bit unsigned integer that MUST represent the maximum number of bytes of data that may be returned in the **Data** field of the SMB_COM_TRANSACTION response to the command.

2.5.7.1.2 RAP NetPrintQEnumResponse

The RAPOutParams structure for the NetPrintQEnumResponse is as follows:



EntriesReturned (2 bytes): A 16-bit unsigned integer that MUST represent the number of fixed-size PrintQueue2 data structures returned in the **Data** field of the SMB_COM_TRANSACTION response to NetPrintQEnumRequest.

EntriesAvailable (2 bytes): A 16-bit unsigned integer that MUST represent the number of print queues that are available on the server.

For error conditions and error responses, see section 3.2.5.3.

If the **InfoLevel** specified in the NetPrintQEnumRequest is 2, and the response is not an error, the **RAPOutData** field of the SMB_COM_TRANSACTION response MUST be filled with an array of **EntriesReturned** PrintQueue2 data structure. Immediately following each PrintQueue2 structure, the **RAPOutData** field MUST also contain as many PrintJobInfo0 structures as are represented in the **PrintJobCount** field in the corresponding PrintQueue2 structure.

2.5.7.2 NetPrintQGetInfo Command

The NetPrintQGetInfo command specifies that the server is to return information on the named print queue on the server.

2.5.7.2.1 RAP NetPrintQGetInfoRequest

The fields in the NetPrintQGetInfoRequest message MUST be set as follows:

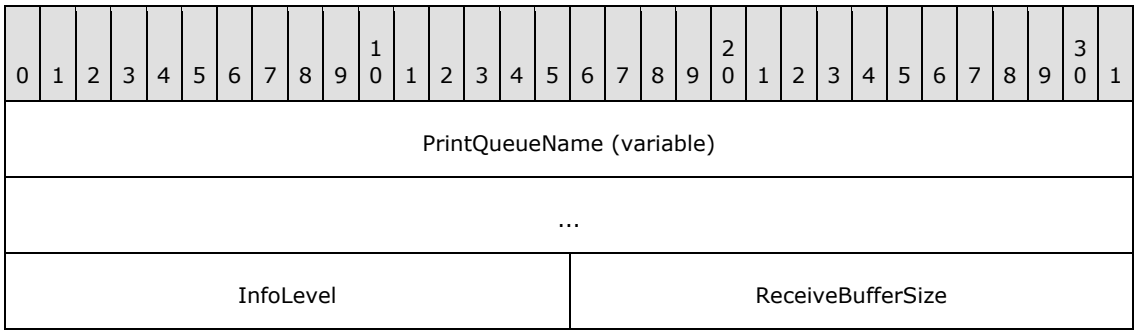
RAPOpcode: MUST be set to 0x0046.

ParamDesc: MUST be set to "zWrLh".

DataDesc: If InfoLevel is set to 0x0002, this MUST be set to "B13BWWWzzzzzWN"; if the InfoLevel is set to 0x0003, this MUST be set to "zWWWWzzzzWWzzl".

AuxDesc: If InfoLevel is set to 0x0002, this MUST be set to "WB21BB16B10zWWzDDz". If InfoLevel is set to 0x0003, this field MUST NOT be present.

RAPParams: The RAPParams structure MUST be as follows:



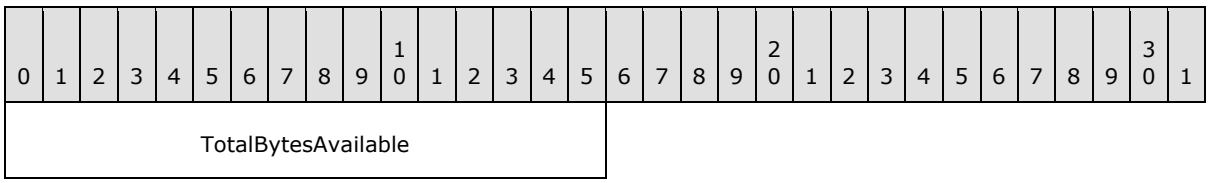
PrintQueueName (variable): A null-terminated ASCII string that MUST specify the name of the print queue to retrieve.

InfoLevel (2 bytes): A 16-bit unsigned integer that MUST specify the information level for NetPrintQGetInfoRequest.

ReceiveBufferSize (2 bytes): A 16-bit unsigned integer that MUST represent the maximum number of bytes of data that may be returned in the **Data** field of the SMB_COM_TRANSACTION response to the command.

2.5.7.2.2 RAP NetPrintQGetInfoResponse

The RAPOutParams structure responds to the NetPrintQGetInfo command with the following.



TotalBytesAvailable (2 bytes): A 16-bit unsigned integer that MUST represent the number of bytes required to hold the information request for the named print queue.

If the InfoLevel specified in the NetPrintQGetInfoRequest is 0, and the response is not an error, the **RAPOutData** field of the SMB_COM_TRANSACTION response MUST be filled with a PrintQueue0 structure. For rules on how to initialize the data structures, see section 3.2.5.4.

If the InfoLevel specified in NetPrintQGetInfoRequest is 2, and the response is not an error, the **RAPOutData** field of the SMB_COM_TRANSACTION response MUST be filled with a PrintQueue2 structure. Immediately following the PrintQueue2 structure, the **RAPOutData** field MUST contain as many PrintJobInfo0 structures as are represented in the **PrintJobCount** field in the PrintQueue2 structure. For rules on how to initialize the data structures, see section 3.2.5.4.

If the InfoLevel specified in NetPrintQGetInfoRequest is 3, and the response is not an error, the **RAPOutData** field of the SMB_COM_TRANSACTION response MUST be filled with a PrintQueue3 structure. Immediately following the PrintQueue3 structure, the **RAPOutData** field MUST contain as many PrintJobInfo3 structures as are represented in the **PrintJobCount** field in the PrintQueue3 structure. For rules on how to initialize the data structures, see section 3.2.5.4.

2.5.7.3 NetPrintJobSetInfo Command

The NetPrintJobSetInfo command specifies that the server MUST modify information on the specified **print job**.

2.5.7.3.1 RAP NetPrintJobSetInfoRequest

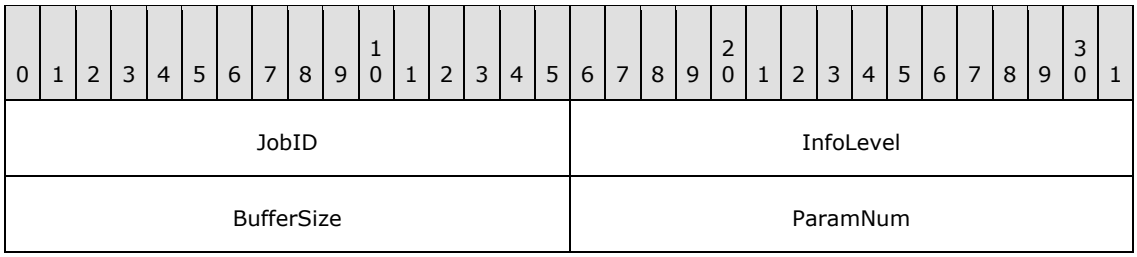
The fields in the NetPrintJobSetInfoRequest message MUST be set as follows:

RAPOpcode: MUST be set to 0x0093.

ParamDesc: MUST be set to "WWsTP".

DataDesc: MUST be set to "WB21BB16B10zWWzDDz".

RAPParams: The RAPParams structure MUST be as follows:



JobID (2 bytes): A 16-bit unsigned integer that MUST contain the **job ID** of the job to modify.

InfoLevel (2 bytes): A 16-bit unsigned integer that MUST specify the information level for the NetPrintJobSetInfoRequest. This MUST be set to 0x0001.

BufferSize (2 bytes): A 16-bit unsigned integer that MUST represent the size of the **RAPInData** structure.

ParamNum (2 bytes): A 16-bit unsigned integer from the table below that MUST specify what aspect of the print job is being modified:

The **Data** field of the SMB_COM_TRANSACTION request MUST be present and set as follows.

Value	Meaning
JobNum 0x0001	A 16-bit integer.
UserName 0x0002	A null-terminated ASCII string.
NotifyName 0x0003	A null-terminated ASCII string.
DataType 0x0004	A null-terminated ASCII string.
ParametersString 0x0005	A null-terminated ASCII string.

Value	Meaning
JobPosition 0x0006	A 16-bit integer.
JobStatus 0x0007	A 16-bit integer.
JobStatus 0x0008	A null-terminated ASCII string.
TimeSubmitted 0x0009	A 32-bit integer.
JobSize 0x000A	A 32-bit integer.
JobComment 0x000B	A null-terminated ASCII string.

RAPInData: This field MUST be based on the **ParamNum** value in the incoming application request.

For example, if the incoming application request sets the **ParamNum** value to 0x000B, the **RAPInData** field MUST be set to a null-terminated ASCII string that represents the new value for the **JobComment** field in the print job specified by the job ID incoming parameter.

2.5.7.3.2 RAP NetPrintJobSetInfoResponse

The **RAPOutParams** field and the **RAPOutData** field of the Remote Administration Protocol response to the NetPrintJobSetInfo command MUST be empty.

2.5.7.4 NetPrintJobGetInfo Command

The NetPrintJobGetInfo command specifies that the server MUST return information on the specified print job.

2.5.7.4.1 RAP NetPrintJobGetInfoRequest

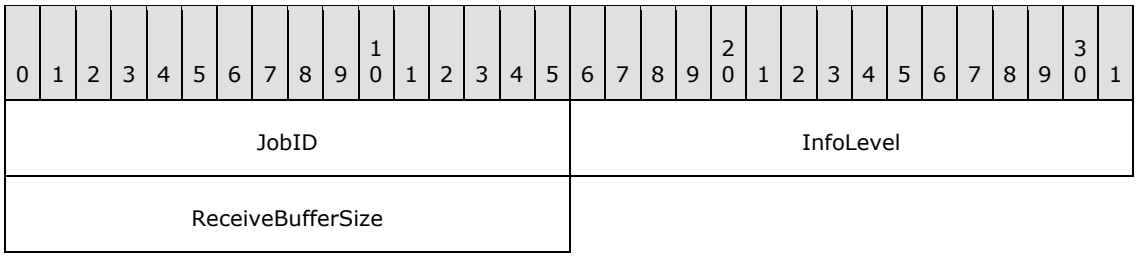
The fields in the NetPrintJobGetInfoRequest message MUST be set as follows:

RAPOpcode: MUST be set to 0x004D.

ParamDesc: MUST be set to "WWrLh".

DataDesc: MUST be set to "WWzWWDDzzzzzzzzzzlz".

RAPParams: The RAPParams structure MUST be as follows:



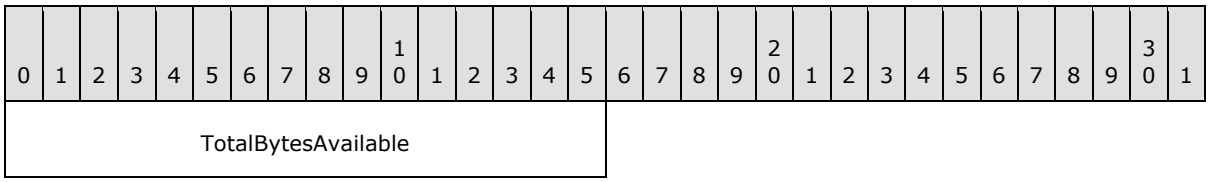
JobID (2 bytes): A 16-bit unsigned integer that MUST contain the job ID of the job to modify.

InfoLevel (2 bytes): A 16-bit unsigned integer that MUST specify the information level for the NetPrintJobGetInfoRequest. This MUST be set to 0x0001.

ReceiveBufferSize (2 bytes): A 16-bit unsigned integer that MUST represent the maximum number of bytes of data that may be returned in the **Data** field of the SMB_COM_TRANSACTION response to the command.

2.5.7.4.2 RAP NetPrintJobGetInfoResponse

The **RAPOutParams** response to the NetPrintJobGetInfo command is as follows.



TotalBytesAvailable (2 bytes): A 16-bit unsigned integer that MUST represent the number of bytes required to hold the requested print job information.

If the InfoLevel of the NetPrintJobGetInfoRequest is 0x0000, the **RAPOutData** of the Remote Administration Protocol response MUST be set to the PrintJobInfo0 structure for the specified job ID.

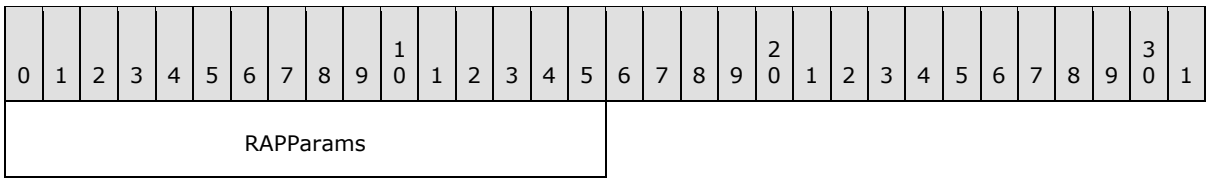
If the InfoLevel of the NetPrintJobGetInfoRequest is 0x0003, the **RAPOutData** of the Remote Administration Protocol response MUST be set to the PrintJobInfo3 structure for the specified job ID.

2.5.7.5 NetPrintJobPause Command

The NetPrintJobPause command specifies that the server MUST pause the specified print job.

2.5.7.5.1 RAP NetPrintJobPauseRequest

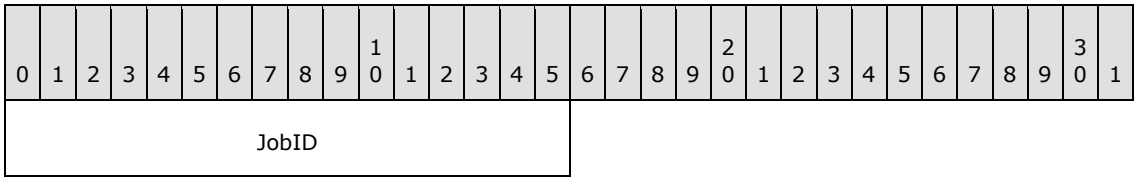
The fields in the NetPrintJobPauseRequest message MUST be set as follows:



RAPOpcode: MUST be set to 0x0052.

ParamDesc: MUST be set to "W".

RAPParams (2 bytes): The RAPParams structure MUST be as follows:



JobID (2 bytes): A 16-bit unsigned integer that MUST represent the job ID of the print job to be paused.

2.5.7.5.2 RAP NetPrintJobPauseResponse

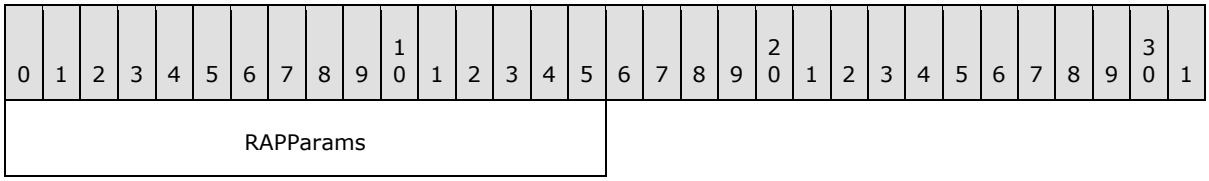
The **RAPOutParams** field and the **RAPOutData** field of the SMB_COM_TRANSACTION response to the NetPrintJobPause command MUST be empty.

2.5.7.6 NetPrintJobContinue Command

The NetPrintJobContinue command specifies that the server MUST continue the specified print job.

2.5.7.6.1 RAP NetPrintJobContinueRequest

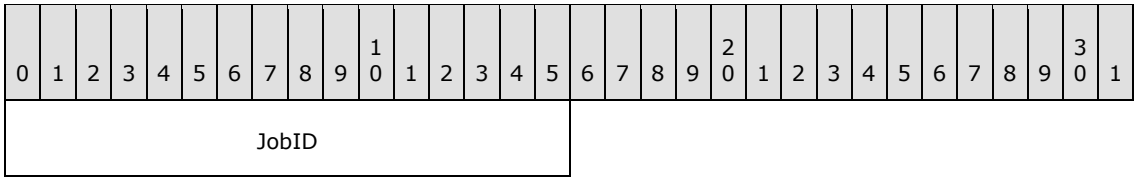
The fields in the NetPrintJobContinueRequest message MUST be set as follows:



RAPOpcode: MUST be set to 0x0053.

ParamDesc: MUST be set to "W".

RAPParams (2 bytes): This structure MUST be set as follows:



JobID (2 bytes): A 16-bit unsigned integer that MUST represent the job ID of the print job to be continued.

2.5.7.6.2 RAP NetPrintJobContinueResponse

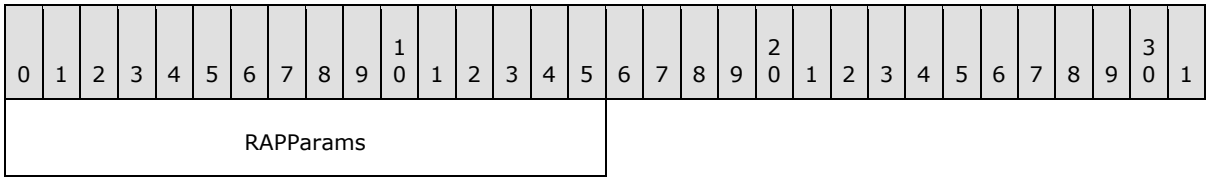
The **RAPOutParams** field and the **RAPOutData** field of the SMB_COM_TRANSACTION response to the NetPrintJobContinue command MUST be empty.

2.5.7.7 NetPrintJobDelete Command

The NetPrintJobDelete command specifies that the server is to delete the specified print job.

2.5.7.7.1 RAP NetPrintJobDeleteRequest

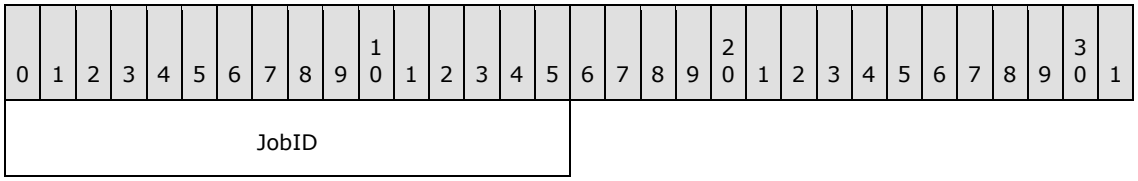
The fields in the NetPrintJobDeleteRequest message MUST be set as follows:



RAPOpcode: MUST be set to 0x0051.

ParamDesc: MUST be set to "W".

RAPParams (2 bytes): This structure MUST be set as follows:



JobID (2 bytes): A 16-bit unsigned integer that MUST represent the job ID of the print job to be deleted.

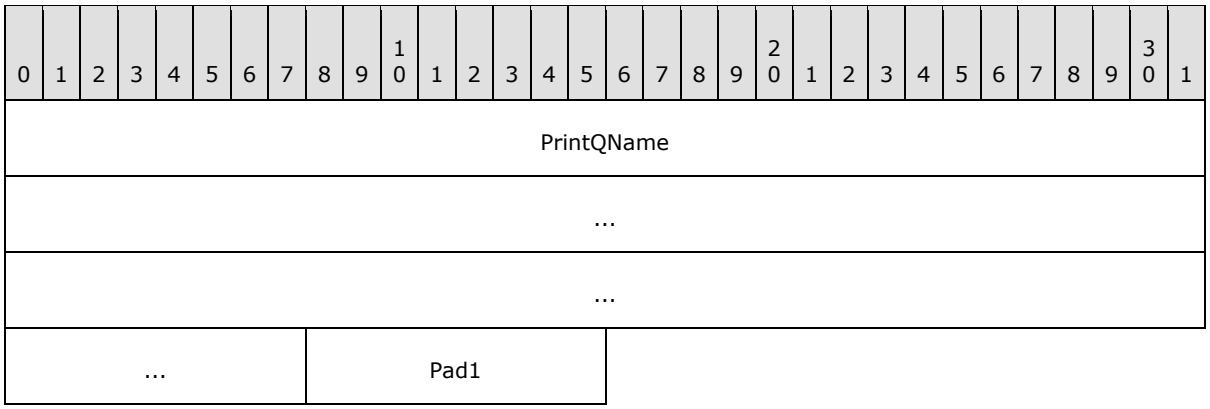
2.5.7.7.2 RAP NetPrintJobDeleteResponse

The **RAPOutParams** field and the **RAPOutData** field of the SMB_COM_TRANSACTION response to the NetPrintJobDelete command MUST be empty.

2.5.7.8 RAP Print Response Structures

2.5.7.8.1 RAP PrintQueue0

The data field in the response to a NetPrintQGetInfo command MUST consist of an array of structures as follows:

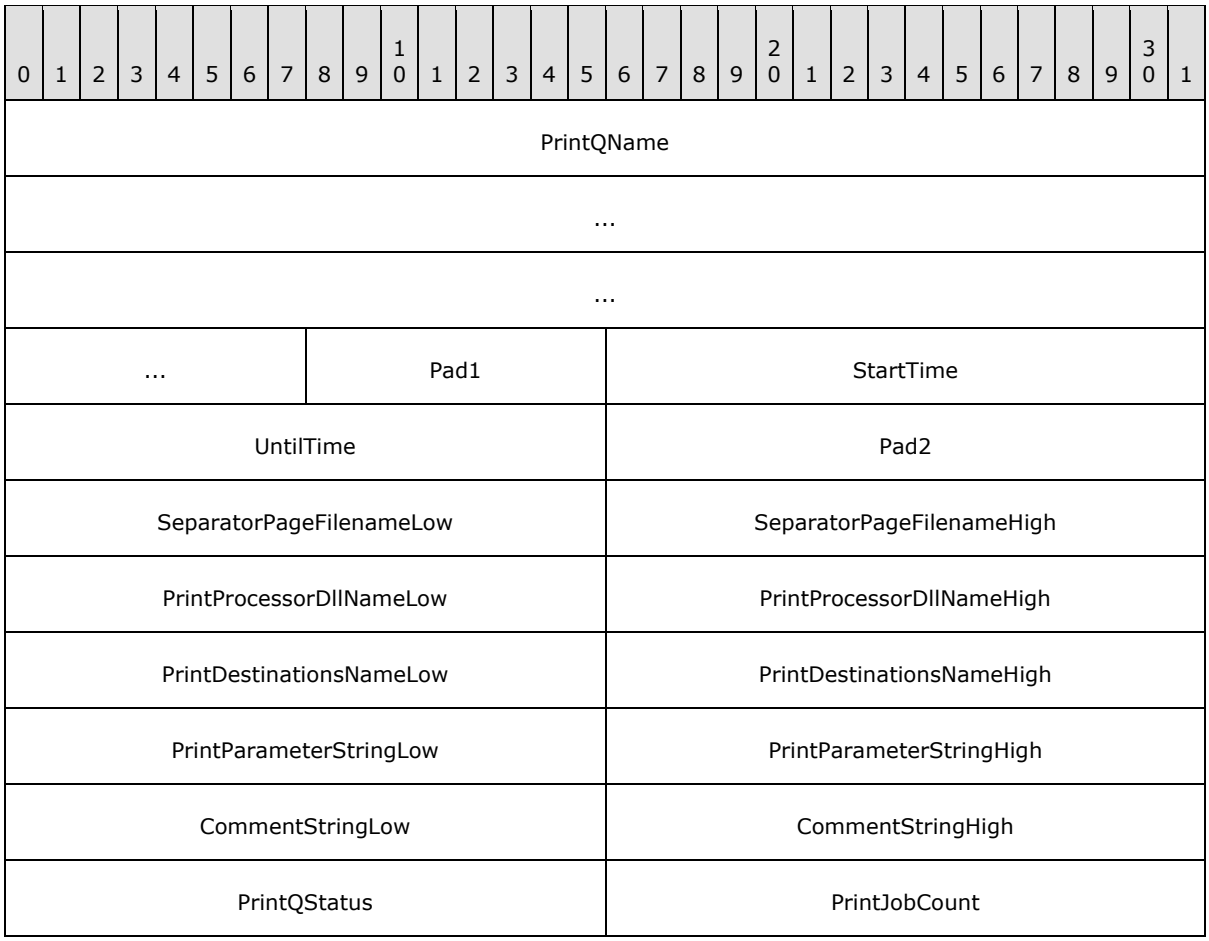


PrintQName (13 bytes): This field MUST contain an ASCII null-terminated name of the print queue that MUST be padded to 13 bytes with ASCII null characters (0x00).

Pad1 (1 byte): Pad byte. Set to an arbitrary value on send, and MUST be ignored on receipt.

2.5.7.8.2 RAP PrintQueue2 Structure

The data field (see section 2.5.2) in the response to a NetPrintQEnum command or a NetPrintQGetInfo command MUST consist of an array of structures as follows:



PrintQName (13 bytes): This field MUST contain an ASCII null-terminated name of the print queue that MUST be padded to 13 bytes with ASCII null characters (0x00).

Pad1 (1 byte): A pad byte that is set to an arbitrary value on send and that MUST be ignored on receipt.

StartTime (2 bytes): A 16-bit unsigned integer that MUST represent the print queue start time (in minutes since midnight) in the local time zone of the server. A print queue accepts jobs, but only prints the jobs after the **StartTime** has elapsed. The **StartTime** field MUST be less than 1,440 minutes.

UntilTime (2 bytes): A 16-bit unsigned integer that MUST represent the print queue stop time. After this time, jobs are accepted but are not printed. This value is expressed (in minutes since midnight) in the local time zone of the server. The **UntilTime** field MUST be less than 1,440 minutes.

Pad2 (2 bytes): Pad bytes that are set to an arbitrary value on send and that MUST be ignored on receipt.

SeparatorPageFilenameLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify the local

file name that contains the **printer separator page**. If no printer separator page is configured, this value MUST be an empty string.

Before using this value, a Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **SeparatorPageFilenameLow** value, and then use that result as the offset within the response.

This file name is for informational purposes only; clients MUST NOT take any action other than to display or log it.

SeparatorPageFilenameHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

PrintProcessorDllNameLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify the file name of the DLL that contains the print processor for this print queue.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **PrintProcessorDllNameLow** value, and then use that result as the offset within the response. This file name is for informational purposes only; a client MUST NOT take any action other than to display or log it.

PrintProcessorDllNameHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

PrintDestinationsNameLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that lists the **print destinations** for this print queue. Each print destination is separated by an ASCII space character (0x20).

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **PrintDestinationsLow** value, and then use that result as the offset within the response.

This field is for informational purposes only; a client MUST NOT take any action other than to display or log it.

PrintDestinationsNameHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

PrintParameterStringLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify parameters for this print queue.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **PrintParameterStringLow** value, and then use that result as the offset within the response.

This field is for informational purposes only; a client MUST NOT take any action other than to display or log it.

PrintParameterStringHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

CommentStringLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify the print queue.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **CommentStringLow** value, and then use that result as the offset within the response.

CommentStringHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

PrintQStatus (2 bytes): An enumeration that MUST specify the status of the print queue. The following values MUST be used for the **PrintQStatus** field.

Value	Meaning
PRQ_ACTIVE 0x0000	The queue is accepting print jobs.
PRQ_PAUSE 0x0001	The queue is paused.
PRQ_ERROR 0x0002	The queue is in an error state.
PRQ_PENDING 0x0003	The queue is marked for deletion.

PrintJobCount (2 bytes): The number of PrintJobInfo0 structures that follow the PrintQueue2 structure.

For more information on the Remote Administration Protocol PrintQueue2 structure, see [RYAN] page 409.

2.5.7.8.3 RAP PrintQueue3 Structure

The data field in the response to a NetPrintQEnum command or a NetPrintQGetInfo command MUST consist of the following 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
PrintQueueNameLow																PrintQueueNameHigh															
Priority																StartTime															
UntilTime																Pad															
SeparatorPageFilenameLow																SeparatorPageFilenameHigh															
PrintProcessorDllNameLow																PrintProcessorDllNameHigh															
PrintParameterStringLow																PrintParameterStringHigh															
PrintQStatus																PrintJobCount															
CommentStringLow																CommentStringHigh															
PrintersLow																PrintersHigh															
DriverNameLow																DriverNameHigh															
PrintDriverDataLow (optional)																PrintDriverDataHigh (optional)															

PrintQueueNameLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that contains the name of the print queue.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **PrintQueueNameLow** value, and then use that result as the offset within the response.

This field is for informational purposes only; a client MUST NOT take any action other than to display or log it.

PrintQueueNameHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

Priority (2 bytes): A 16-bit unsigned integer that MUST specify the priority of the print queue. Valid values are 0x0001 (highest) to 0x0009 (lowest). When two printer queues print to the same printer, the print jobs from the queue with the higher priority print first.

StartTime (2 bytes): A 16-bit unsigned integer that MUST represent the print queue start time (in minutes since midnight) in the local time zone of the server. A print queue accepts jobs but only prints the jobs after the **StartTime** value has elapsed. The **StartTime** field MUST be less than 1,440 minutes.

UntilTime (2 bytes): A 16-bit unsigned integer that MUST represent the print queue stop time. After this time, jobs are accepted but are not printed. This value is expressed (in minutes since midnight) in the local time zone of the server. The **UntilTime** field MUST be less than 1,440 minutes.

Pad (2 bytes): Pad bytes that are set to an arbitrary value on send and that MUST be ignored on receipt.

SeparatorPageFilenameLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that contains the file name that contains the printer separator page for the share.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **SeparatorPageFilenameLow** value, and then use that result as the offset within the response. This field is for informational purposes only; a client MUST NOT take any action other than to display or log it.

SeparatorPageFilenameHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

PrintProcessorDllNameLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that contains the file name of the DLL that contains the print processor for this print queue.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **PrintProcessorDllNameLow** value, and then use that result as the offset within the response.

This field is for informational purposes only; a client MUST NOT take any action other than to display or log it.

PrintProcessorDllNameHigh (2 bytes): Unused. Set to any arbitrary value on send and MUST be ignored on receipt.

PrintParameterStringLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that specifies parameters for this print queue.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **PrintParameterStringLow** value, and then use that result as the offset within the response.

This field is for informational purposes only; a client MUST NOT take any action other than to display or log it.

PrintParameterStringHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

PrintQStatus (2 bytes): An enumeration that specifies the status of the print queue. Valid values are the same as those specified in section 2.5.7.8.2.

PrintJobCount (2 bytes): A 16-bit unsigned integer that MUST represent the number of PrintJobInfo0 structures that follow the PrintQueue2 structure.

CommentStringLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify the print queue.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **CommentStringLow** value.

CommentStringHigh (2 bytes): Unused. Set to any arbitrary value on send and MUST be ignored on receipt.

PrintersLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) that specifies the name of the printer. Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **PrintersLow** value, and then use that result as the offset within the response. This field is for informational purposes only; a client MUST NOT take any action other than to display or log it.

PrintersHigh (2 bytes): Unused. Set to any arbitrary value on send and MUST be ignored on receipt.

DriverNameLow (2 bytes): An optional 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) that specifies the default device driver for this queue.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **DriverNameLow** value, and then use that result as the offset within the response.

This field is for informational purposes only; a client MUST NOT take any action other than to display or log it. If the **DriverNameLow** field and the **DriverNameHigh** field are both 0x0000, the **DriverName** field is not present.

DriverNameHigh (2 bytes): Unused. This value MUST be set to 0x0000 on send and MUST be ignored on receipt. The **DriverNameHigh** portion is not used because the total offset cannot be more than the maximum value of **DriverNameLow** due to packet length limitations.

PrintDriverDataLow (2 bytes): An optional 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that contains driver-specific binary data.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **PrintDriverDataLow** value, and then use that result as the offset within the response. The first two bytes of this buffer contain a 16-bit integer that represents the length of the buffer.

This field is for informational purposes only; a client MUST NOT take any action other than display or log it. If the **PrintDriverDataLow** field and the **PrintDriverDataHigh** field are both 0x0000, the **DriverName** field is not present.

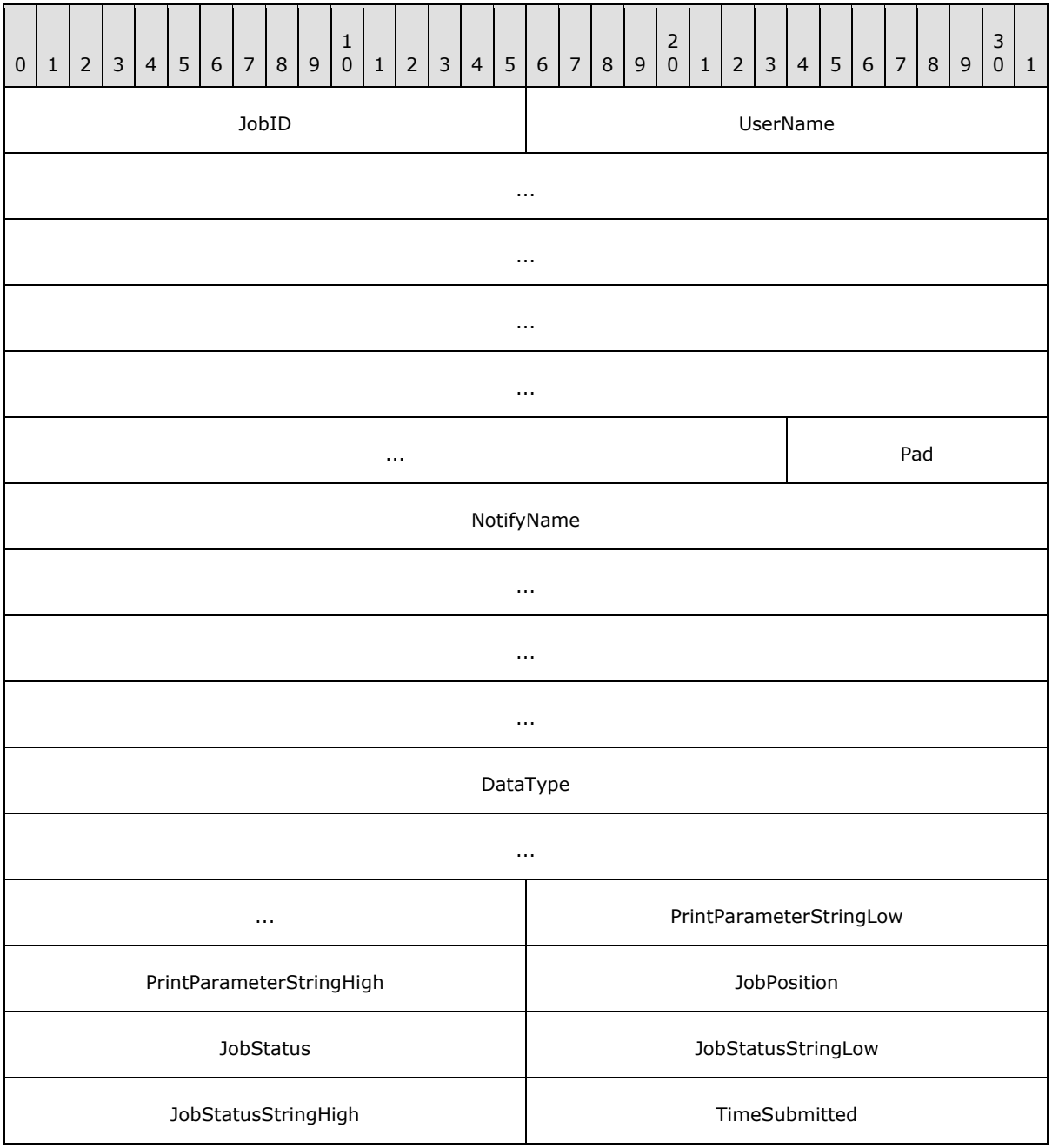
PrintDriverDataHigh (2 bytes): Unused. This value MUST be set to 0x0000 on send and MUST be ignored on receipt. The **PrintDriverDataHigh** portion is not used because the total offset cannot be more than the maximum value of **PrintDriverDataLow** due to packet length limitations.

This field is present if, and only if, the **PrinterDriverDataLow** field is also present.

For more information on the PrintQueue3 structure, see [CIFSPRINT] section 6.1.1 and [RYAN] page 409.

2.5.7.8.4 RAP PrintJobInfo0 Structure

The PrintJobInfo0 structure is returned by the NetPrintQEnum command and by the NetPrintJobGetInfo command:



...	JobSize
...	JobCommentStringLow
JobCommentStringHigh	

JobID (2 bytes): A 16-bit unsigned integer that MUST represent the job ID of the print job.

UserName (21 bytes): Null-terminated ASCII text that MUST contain the name of the user that submitted the job. This field MUST be padded with null characters to 21 bytes in length.

Pad (1 byte): MUST be null to pad the buffer to a 16-bit boundary.

NotifyName (16 bytes): Null-terminated ASCII string that MUST contain the name of the computer that SHOULD be notified when this print job completes. This field MUST be padded with null characters to 16 bytes in length.

DataType (10 bytes): Null-terminated ASCII string that MUST specify an implementation-specific data type assignment of the form TYPE=xxx. This field MUST be padded with null characters to 10 bytes in length.

PrintParameterStringLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify the parameters for this print job. Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **PrintParametersStringLow** value, and then use that result as the offset within the response.

PrintParameterStringHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

JobPosition (2 bytes): A 16-bit unsigned integer that MUST specify the position of this job in the queue. A value of 0x0001 indicates that this job is the next job to print.

JobStatus (2 bytes): A 16-bit unsigned integer that MUST specify the status of this job in the print queue. **JobStatus** MUST be one of the values in the following table:

Value	Meaning
PRJ_QS_QUEUED 0x0000	Job is in the queue.
PRJ_QS_PAUSED 0x0001	Job is in the queue but paused.
PRJ_QS_SPOOLING 0x0002	Job is being written to the spooler queue.
PRJ_QS_PRINTING 0x0003	Job is being printed.

JobStatusStringLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the

response block (as specified in section 2.5.10) and that describes the status of this print job. Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **JobCommentStringLow** value, and then use that result as the offset within the response.

JobStatusStringHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

TimeSubmitted (4 bytes): A 32-bit unsigned integer that MUST specify the time that the print job was submitted (in seconds since midnight January 1, 1970) in the local time zone of the server.

JobSize (4 bytes): A 32-bit unsigned integer that MUST specify the size of the print job in bytes.

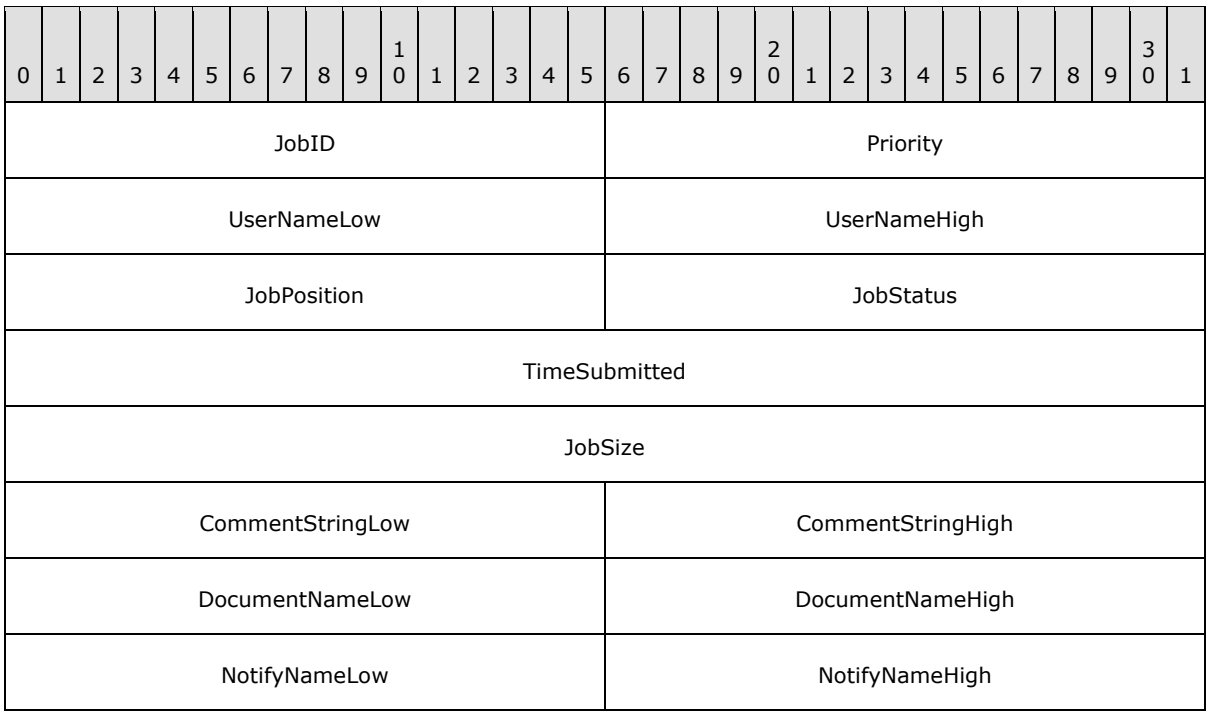
JobCommentStringLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that describes this print job. Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **JobCommentStringLow** value, and then use that result as the offset within the response.

JobCommentStringHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

For more information on the PrintJobInfo0 structure, see [RYAN] page 421.

2.5.7.8.5 RAP PrintJobInfo3 Structure

The PrintJobInfo3 structure is returned by the NetPrintJobGetInfo command and has the following fields:



DataTypeLow	DataTypeHigh
PrintParameterStringLow	PrintParameterStringHigh
StatusStringLow	StatusStringHigh
QueueNameLow	QueueNameHigh
PrintProcessorNameLow	PrintProcessorNameHigh
DriverNameLow	DriverNameHigh
DriverDataOffsetLow	DriverDataOffsetHigh
PrinterNameOffsetLow	PrinterNameOffsetHigh

JobID (2 bytes): A 16-bit unsigned integer that MUST represent the job ID of the print job.

Priority (2 bytes): A 16-bit unsigned integer that MUST represent the priority of the print job. If the value is 0x0000, the priority of the print queue determines the job priority. Other valid values are between 0x0001 and 0x0063, inclusive. When two printer queues print to the same printer, the print jobs from the queue with the higher priority print first.

UserNameLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify the user name that submitted this print job.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **UserNameLow** value, and then use that result as the offset within the response.

UserNameHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

JobPosition (2 bytes): A 16-bit unsigned integer that MUST specify the position of this job in the queue. A value of 0x0001 indicates that this job is the next job to print.

JobStatus (2 bytes): An enumeration that MUST specify the status of this job in the print queue. Its value MUST be as specified in section 2.5.7.8.4.

TimeSubmitted (4 bytes): A 32-bit unsigned integer that MUST specify the time that the print job was submitted (in seconds since midnight January 1, 1970) in the local time zone of the server.

JobSize (4 bytes): A 32-bit unsigned integer that MUST specify the size, in bytes, of the print job.

CommentStringLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the

response block (as specified in section 2.5.10) and that MUST specify a string that describes the print job.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **CommentStringLow** value, and then use that result as the offset within the response.

CommentStringHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

DocumentNameLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify the name of the document.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **DocumentNameLow** value, and then use that result as the offset within the response.

DocumentNameHigh (2 bytes): Unused. Set to any arbitrary value on send and MUST be ignored on receipt.

NotifyNameLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify a computer name that is notified when the status of this print job changes.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **NotifyNameLow** value, and then use that result as the offset within the response.

NotifyNameHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

DataTypeLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify an implementation-specific type string TYPE=xxx.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **DataTypeLow** value, and then use that result as the offset within the response. For more information on the **DataType** field, see [RAP] page 421.

DataTypeHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

PrintParameterStringLow (2 bytes): A 16-bit integer representing the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify the implementation-specific parameters for this print job.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **PrintParameterStringLow** value, and then use that result as the offset within the response.

PrintParameterStringHigh (2 bytes): Unused. Set to any arbitrary value on send and MUST be ignored on receipt.

StatusStringLow (2 bytes): A 16-bit integer representing the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify the status of this print job.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **StatusStringLow** value, and then use that result as the offset within the response.

StatusStringHigh (2 bytes): Unused. Set to any arbitrary value on send and MUST be ignored on receipt.

QueueNameLow (2 bytes): A 16-bit integer representing the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify the name of the print queue that contains this print job.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **QueueNameLow** value, and then use that result as the offset within the response.

QueueNameHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

PrintProcessorNameLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string allocated in the response block (see section 2.5.10) that MUST specify the print processor for this print job.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **PrintProcessorNameLow** value, and then use that result as the offset within the response.

PrintProcessorNameHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

DriverNameLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify the implementation-specific name of the driver for this print job.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **DriverNameLow** value, and then use that result as the offset within the response.

DriverNameHigh (2 bytes): Unused. Set to an arbitrary value on send and MUST be ignored on receipt.

DriverDataOffsetLow (2 bytes): An optional 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that contains driver-specific binary data.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **DriverDataOffsetLow** value, and

then use that result as the offset within the response. The first two bytes of this buffer contain a 16-bit, unsigned integer that represents the length of the buffer.

This field is for informational purposes only; a client MUST NOT take any action other than to display or log it. If the **PrintDriverDataLow** field and the **PrintDriverDataHigh** field are both set to 0x0000, the **DriverName** field is not present.

DriverDataOffsetHigh (2 bytes): Unused. This value MUST be set to 0x0000 on send and MUST be ignored on receipt. The **DriverDataOffsetHigh** portion is not used because the total offset cannot be more than the maximum value of **DriverDataOffsetLow** due to packet length limitations.

PrinterNameOffsetLow (2 bytes): A 16-bit unsigned integer that MUST represent the offset, in bytes, from the start of the response to a null-terminated ASCII string that is allocated in the response block (as specified in section 2.5.10) and that MUST specify the name of the printer associated with this print job.

Before using this value, the Remote Administration Protocol client MUST subtract the **Converter** field, as specified in section 2.5.2, from the **PrinterNameOffsetLow** value, and then use that result as the offset within the response.

PrinterNameOffsetHigh (2 bytes): Unused. Set to any arbitrary value on send and MUST be ignored on receipt.

For more information on the PrintJobInfo3 structure, see [RYAN] page 421.

2.5.8 RAP User Commands

2.5.8.1 NetUserPasswordSet2 Command

The NetUserPasswordSet2 command specifies that the server is to change the password of the indicated user.

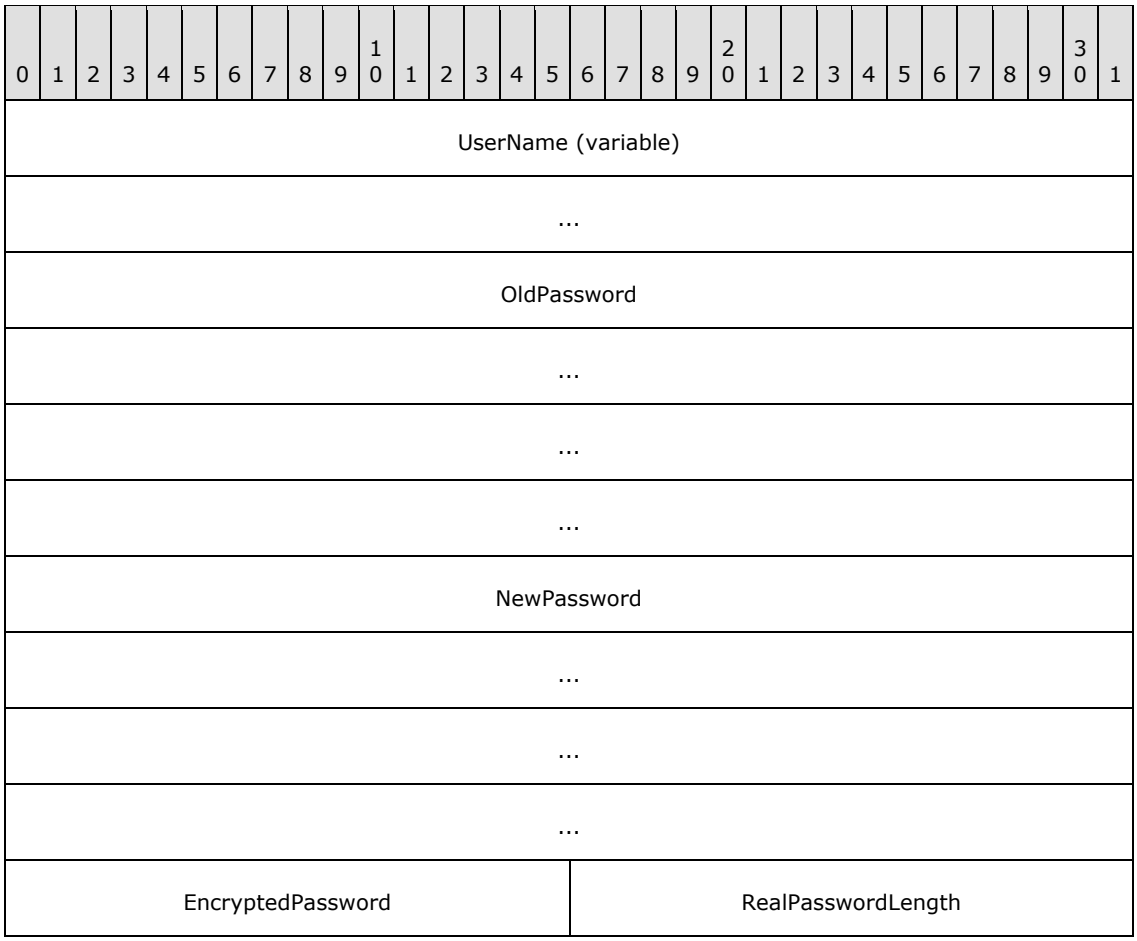
2.5.8.1.1 RAP NetUserPasswordSet2Request

The fields in the NetUserPasswordSet2Request message MUST be set as follows:

RAPOpcode: MUST be set to 0x0073.

ParamDesc: MUST be set to "zb16b16WW".

RAPParams: The **RAPParams** structure MUST be as follows:



UserName (variable): A null-terminated ASCII string that MUST specify the name of the user whose password is being changed.

OldPassword (16 bytes): A 16-byte, null-terminated ASCII string padded with zeros that MUST contain the user's current password.

NewPassword (16 bytes): A 16-byte, null-terminated ASCII string padded with zeros that MUST contain the user's new password.

EncryptedPassword (2 bytes): A 16-bit, unsigned integer that MUST specify if the **OldPassword** and **NewPassword** fields are encrypted. If set to 0x0000, the fields are not encrypted; if not 0, the fields are encrypted.

RealPasswordLength (2 bytes): A 16-bit, unsigned integer that MUST specify the actual length of the **NewPassword** field.<9>

2.5.8.1.2 RAP NetUserPasswordSet2Response

The **RAPOutParams** field and the **RAPOutData** field of the SMB_COM_TRANSACTION response to the NetUserPasswordSet2 command MUST be empty.

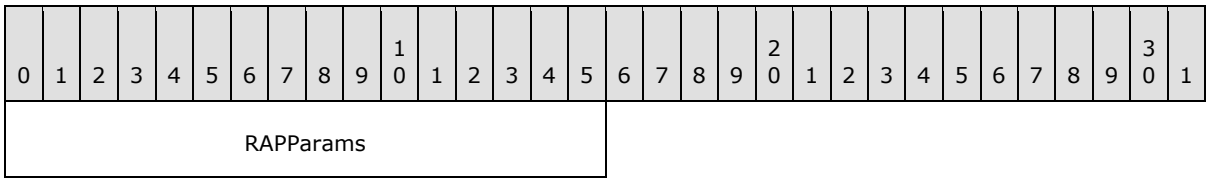
2.5.9 RAP Time Commands

2.5.9.1 NetRemoteTOD Command

The NetRemoteTOD command specifies that the server is to return its current time information.

2.5.9.1.1 RAP NetRemoteTODRequest

The fields in the NetRemoteTODRequest message MUST be set as follows:

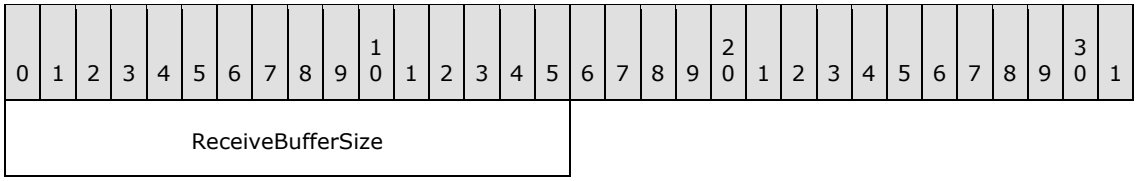


RAPOpcode: MUST be set to 0x005B.

ParamDesc: MUST be set to "rL".

DataDesc: MUST be set to "DDBBBBWWBBWB".

RAPParams (2 bytes): The **RAPParams** structure MUST be as follows:



ReceiveBufferSize (2 bytes): A 16-bit, unsigned integer that MUST represent the maximum number of bytes of data that may be returned in the **Data** field of the SMB_COM_TRANSACTION response to the command.

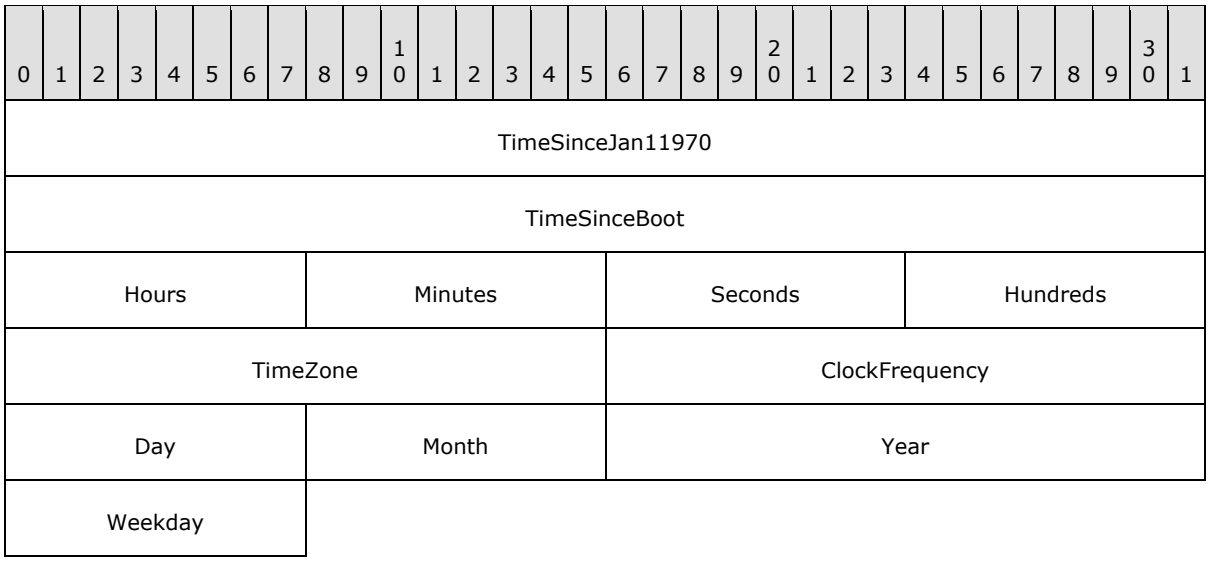
2.5.9.1.2 RAP NetRemoteTODResponse

If the **Win32ErrorCode** specified in the response to the NetRemoteTODRequest is either ERROR_SUCCESS (0x0000) or ERROR_MORE_DATA (0x00EA), the **RAPOutData** field of the SMB_COM_TRANSACTION response MUST be filled with a TimeOfDayInfo structure. If the **Win32ErrorCode** is any other value, the SMB_COM_TRANSACTION response MUST be empty.

2.5.9.2 RAP Time Structures

2.5.9.2.1 RAP TimeOfDayInfo Structure

The data section of the response to a NetRemoteTOD command MUST be as follows:



TimeSinceJan11970 (4 bytes): A 32-bit, unsigned integer that MUST be the number of seconds since midnight January 1, 1970, Coordinated Universal Time (UTC).

TimeSinceBoot (4 bytes): A 32-bit, unsigned integer that MUST be the time, in milliseconds, since computer system reset.

Hours (1 byte): An 8-bit, unsigned integer that MUST be the current hour of the day in the server's local time zone. Valid values are from 0x00 to 0x17, inclusive.

Minutes (1 byte): An 8-bit, unsigned integer that MUST be the current minute in the server's local time zone. Valid values are from 0x00 to 0x3B, inclusive.

Seconds (1 byte): An 8-bit, unsigned integer that MUST be the current second in the server's local time zone. Valid values are from 0x00 to 0x3B, inclusive.

Hundreds (1 byte): An 8-bit, unsigned integer that MUST be the hundredth of a second in the server's local time zone. Valid values are from 0x00 to 0x63, inclusive.

TimeZone (2 bytes): A 16-bit integer that MUST be the time zone of the server. This value is represented in minutes from UTC. For time zones west of UTC, the value is positive; for time zones east of UTC, the value is negative.

ClockFrequency (2 bytes): A 16-bit, unsigned integer that MUST be the resolution of the clock in 1/10,000 of a second (0.0001 second).

Day (1 byte): An 8-bit, unsigned integer that MUST be the day of the month. Valid values are from 0x01 to 0x1F, inclusive.

Month (1 byte): An 8-bit, unsigned integer that MUST be the month of the year. Valid values are from 0x01 to 0x0C, inclusive.

Year (2 bytes): A 16-bit, unsigned integer that MUST be the year.

Weekday (1 byte): An 8-bit, unsigned integer that MUST be the day of the week. Valid values are from 0x00 to 0x06, inclusive, in which 0x00 is Sunday, 0x01 is Monday, and so on.

2.5.10 RAP Response Data Marshaling

The response for a Remote Administration Protocol command can contain one or more fixed-size items, each of which can contain offsets to variable length data (typically strings), depending on the command. These fixed-size items MUST be returned in the **RAPOutData** field of the SMB_COM_TRANSACTION response that corresponds to the SMB_COM_TRANSACTION request that contained the Remote Administration Protocol request.

The server MUST NOT return more information in the **Data** field of the SMB_COM_TRANSACTION response than is specified in the *ReceiveBufferSize* of the Remote Administration Protocol request. This section uses the term "response buffer" to represent a buffer, whose size is *ReceiveBufferSize*, that will be sent in the **RAPOutData** field of the response.

When a server implementing the Remote Administration Protocol copies the fixed-size items into the response buffer, it copies them beginning at the buffer's first byte. Variable-length data is copied into the response buffer after the fixed-size items.

When a Remote Administration Protocol server copies a fixed-size item to the response buffer, the Remote Administration Protocol server MUST copy the entire structure into the response buffer. If the Remote Administration Protocol server cannot fit the entire data structure into the response buffer, it MUST set the **Win32ErrorCode** in the Remote Administration Protocol response message to ERROR_MORE_DATA, and continue processing items.

If the server cannot fit any of the fixed-size data structures into the response buffer, the Remote Administration Protocol server MUST set the **Win32ErrorCode** in the Remote Administration Protocol response message to ERROR_INSUFFICIENT_BUFFER (0x007A).

When marshaling more than one data structure, the Remote Administration Protocol server MUST pack each response data structure immediately after the previous response data structure.

When marshaling a variable-length string that is pointed to by an offset in the fixed-size section, if the string data fits into the response buffer, the corresponding field in the fixed-size section MUST be set to 0. All strings are encoded in ASCII data and are terminated with a single null character. If the source string is null, then it MUST be marshaled as an empty string consisting of a single null character.

For certain Remote Administration Protocol commands, such as NetPrintQEnum and NetPrintQGetInfo, the fixed-size portion of the response packet also contains auxiliary data structures. For more information on these commands, see [RYAN] page 410. If the Remote Administration Protocol server cannot fit all of the auxiliary structures into the response buffer, it MUST NOT copy any of the data in the fixed-size structure OR the auxiliary data structures to the response buffer.

As an example of this marshaling format, consider the case of a server marshaling a fixed-size data structure that has one or more auxiliary data structures associated with it. In this example, the fixed-size data structure consists of two 16-bit unsigned integers, an unsigned AUXCOUNT value, and an additional 16-bit unsigned integer, while the auxiliary data structure consists of two 32-bit unsigned integers. If the server marshals two instances of the data structure (called Data 1 and Data 2, for example), both of which have three auxiliary data structures associated with it, the server MUST marshal the following values into the response buffer.

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
Data 1 W 1 value																Data 1 W 2 value															
Data 1 AUXCOUNT=3																Data 1 W 3 value															
Data 1 AUX 1 D 1value																															
Data 1AUX 1 D 2 value																															
Data 1 AUX 2 D 1 value																															
Data 1 AUX 2 D 2 value																															
Data 1 AUX 3 D 1 value																															
Data 1 AUX 3 D 2 value																															
Data 2 W 1 value																Data 2 W 2 value															
Data 2 AUXCOUNT=3																Data 2 W 3 value															
Data 2 AUX 1 D 1value																															
Data 2 AUX 1 D 2 value																															
Data 2 AUX 2 D 1 value																															
Data 2 AUX 2 D 2 value																															
Data 2 AUX 3 D 1 value																															
Data 2 AUX 3 D 2 value																															

3 Protocol Details

3.1 RAP Client Details

3.1.1 Abstract Data Model

No state is maintained by the client, so no abstract data model is needed.

3.1.2 Timers

There are no timers required.

3.1.3 Initialization

The Remote Administration Protocol client MUST establish a connection to the server using the pipe name \PIPE\LANMAN, as specified in section 2.1. No initializations are required.

3.1.4 Higher-Layer Triggered Events

There is a one-to-one correspondence between higher-layer triggered events and methods specified in section 2.4. When a higher layer requests a particular action, the associated method MUST be passed to the Remote Administration Protocol with all values as specified by the higher layer. Details of the parameters supplied by the higher layer are specified in the request structure for the command in section 2.5.1. The client MUST propagate the values provided by the calling application and MUST fail the call if the parameters required are not provided, or if an illegal InfoLevel is provided.

The client MUST create a Remote Administration Protocol request message, as specified in section 2.5.5, for the command corresponding to the requested operation. If a value is not specified for **ParamDesc**, **DataDesc**, or **AuxDesc** in the corresponding section, the field MUST NOT be transmitted with the request.

The client MUST then submit an SMB_COM_TRANSACTION request (that MUST contain the Remote Administration Protocol request) to the server.

3.1.4.1 NetShareEnum Command

The client MUST create a NetShareEnumRequest.

3.1.4.2 NetServerGetInfo Command

The client MUST create a NetServerGetInfoRequest.

3.1.4.3 NetPrintQEnum Command

The client MUST create a NetPrintQEnumRequest.

3.1.4.4 NetPrintQGetInfo Command

The client MUST create a NetPrintQGetInfoRequest.

3.1.4.5 NetPrintJobSetInfo Command

The client MUST create a NetPrintJobSetInfoRequest.

3.1.4.6 NetPrintJobGetInfo Command

The client MUST create a NetPrintJobGetInfoRequest.

3.1.4.7 NetPrintJobDelete Command

The client MUST create a NetPrintJobDeleteRequest.

3.1.4.8 NetPrintJobPause Command

The client MUST create a NetPrintJobPauseRequest.

3.1.4.9 NetPrintJobContinue Command

The client MUST create a NetPrintJobContinueRequest.

3.1.4.10 NetRemoteTOD Command

The client MUST create a NetRemoteTODRequest.

3.1.4.11 NetServerEnum2 Command

The client MUST create a NetServerEnum2Request.

3.1.4.12 NetUserPasswordSet2 Command

The client MUST create a NetUserPasswordSet2Request.

3.1.4.13 NetServerEnum3 Command

The client MUST create a NetServerEnum3Request.

3.1.5 Message Processing Events and Sequencing Rules

If the underlying SMB protocol indicates that a response has been successfully received from the server, the values returned in the **Win32ErrorCode** field of the a Remote Administration Protocol response (as well as any response parameters or data) MUST be returned to the calling higher layer.

If the underlying SMB protocol indicates that an error has occurred or that the connection has been disconnected, the error code MUST be returned to the calling higher layer with no response data.

3.1.6 Timer Events

There are no timer events required.

3.1.7 Other Local Events

There are no local events required.

3.2 RAP Server Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to explain how the protocol behaves. This specification does not mandate the internal data structures used by a server to implement the conceptual model as long as their external behavior conforms to the described normative behavior.

3.2.1.1 Global

A Remote Administration Protocol implementation maintains the following data. These data descriptions are provided to explain the protocol's behavior. This specification does not mandate the internal data structures a server uses so long as their external behavior conforms to the described normative behavior.

ShareTable: A table of network shares that the server hosts. The table MUST be uniquely indexed by Share.Name. See section 3.2.1.2.

UserTable: A table of user accounts that exists on the server. The table MUST be uniquely indexed by User.Name.

ServerList: A list of server machines that exist on a network. In most implementations, this list is managed by an outside service, such as the CIFS Browser Protocol (for more information, see [MS-BRWS]). This list MUST be maintained in alphabetical order.

PrintQueueList: A list of printer queues that the server hosts.

3.2.1.2 Share

A share in a **ShareTable** MUST contain the following attributes. For more information about the use of shares, see shares, as specified in [CIFS] section 4.1.4.

Share.Name: : The name of the share. This value MUST be an ASCII string of 80 characters or fewer, not including any null terminator.

Share.Type: The type of the share. This value MUST correspond to the **Type** field, as specified in section 2.5.6.2.2.

Share.Remark: The remark associated with this share. This value MUST be an ASCII string of 48 characters or fewer, not including any null terminator.

3.2.1.3 User

A user in the **UserTable** MUST contain the following attributes:

User.Name: The name of the user.

User.Password: The password of the user. This value MUST be an ASCII string or a secure representation (that is, a hash) of the password string.

3.2.1.4 Server

A server listed in the **ServerList** MUST contain the following attributes:

Server.Name: The textual name of the server. This value MUST be a NetBIOS name (as specified in [RFC1001]).

Server.VersionHigh: The first 8-bit number that represents the most significant byte of the version of the server.

Server.VersionLow: The second 8-bit number that represents the least significant byte of the version of the server.

Server.Type: The type of the server, as defined in section 2.5.5.2.1.

Server.Comment: A comment for the server. This value MUST be an ASCII string of 48 characters or fewer, not including any null terminator.

3.2.1.5 Print Queue

A print queue in **PrintQueueList** MUST contain the following attributes. For more information on print queues, see [MS-RPRN].

PrintQueue.Name: The name of this print queue.

PrintQueue.Priority: The priority of the print queue, as defined in section 2.5.7.8.3.

PrintQueue.StartTime: The time of day (in minutes after midnight) that the printer starts printing; MUST be less than 1,440.

PrintQueue.StopTime: The time of day (in minutes after midnight) that the printer stops printing; MUST be less than 1,440.

PrintQueue.Separator: An ASCII string that MUST specify the implementation-specific separator page for the printer.

PrintQueue.DLL: An ASCII string that MUST specify the implementation-specific print processor DLL for the printer.

PrintQueue.Destination: An ASCII string that MUST specify the implementation-specific set of print destinations for the print queue.

PrintQueue.Parameters: An ASCII string that MUST specify the implementation-specific parameters for the printer associated with this print queue.

PrintQueue.Description: An ASCII string with the description of this print queue.

PrintQueue.PrintJobList: A list of the print jobs that this printer is handling, in the order of processing.

3.2.1.6 Print Job

A print job in **PrintQueue.PrintJobList** MUST contain the following attributes. For more information on print jobs, see [MS-RPRN].

PrintJob.ID: The identifier for the print job.

PrintJob.UserName: The name of the user who submitted the print job.

PrintJob.DataType: An ASCII string with the implementation-specific data type of the print job.

PrintJob.Parameters: An ASCII string with the implementation-specific parameters associated with the print job.

PrintJob.Position: The position of the print job in the **PrintQueue.PrintJobList**. A value of 1 MUST indicate that the print job is the next job to print.

PrintJob.Status: The status of the job in the print queue. Valid values are the same as those specified for **JobStatus** in section 2.5.7.8.4.

PrintJob.Comment: An ASCII string with a comment for the print job.

PrintJob.PrintProvider: A reference to the underlying printing device that is handling the print job. This device MUST allow print jobs to be added, deleted, paused, and continued, and it MUST locally remove print jobs from the print queue as they are processed, as specified in section 3.2.4.1.

Note The above conceptual data can be implemented using a variety of techniques.

3.2.2 Timers

No timers are required to implement the Remote Administration Protocol.

3.2.3 Initialization

The Remote Administration Protocol server MUST register pipe name \PIPE\LANMAN with the local SMB service so that the client behavior, as specified in section 2.1, can enable the client to connect to the Remote Administration Protocol server.

The **ShareTable** MUST be populated based on server configuration. The values MUST be retrieved from a persistent configuration store; for each entry, a share MUST be created and inserted into the ShareTable. The information retrieved from configuration MUST include the following:

- **Share.Name**: MUST be set to the name of the share. This value MUST be 80 characters or fewer.
- **Share.Type**: MUST be set to the type of the share.
- **Share.Remark**: MUST be set to the remark provided for the share. If no remark is present, it MUST be set to the empty string that consists of a single null character.

The UserTable MUST be populated based on server configuration. The values MUST be retrieved from a persistent configuration store; for each entry, a user MUST be created and inserted into the UserTable. The information retrieved from configuration MUST include the following:

- **User.Name**: MUST be set to the name of the user.
- **User.Password**: MUST be set to the password, or the secure representation of the password, based on server implementation.

The ServerList MUST be populated based on server configuration or other services that provide the list of servers on the network such as browsers (for more information, see [MS-BRWS]). Each entry in the ServerList MUST include the following:

- **Server.Name**: MUST be set to the name of the server.
- **Server.Type**: MUST be set to the type of the server.

- **Server.VersionHigh** and **Server.VersionLow**: MUST be set to the version information of the server.
- **Server.Comment**: MUST be set to the comment for the server. If no comment is present, it MUST be set to the empty string, which consists of a single null character.

The PrintQueueList MUST be populated based on server configuration. Each entry in the PrintQueueList MUST include the following:

- **PrintQueue.Name**: MUST be set to the name of the print queue.
- **PrintQueue.Priority**: MUST be set to the priority.
- **PrintQueue.StartTime**: MUST be set to the time at which the printer starts accepting jobs.
- **PrintQueue.StopTime**: MUST be set to the time at which the printer stops accepting jobs.
- **PrintQueue.Separator**, **PrintQueu.DLL**, **PrintQueue.Destination**, and **PrintQueue.Parameters**: MUST be set to implementation-specific values.
- **PrintQueue.Description**: MUST be set to the description of the print queue or, if no description is provided, to the empty string that consists of a single null character.
- **PrintQueue.PrintJobList**: MUST be set to an empty list or, if an implementation supports a persistent store of the **PrintJobList**, it is loaded from the persistent store.

3.2.4 Higher-Layer Triggered Events

None.

3.2.4.1 Local Print Provider Completes a Print Job

When a local print provider completes a print job, it indicates this to the Remote Administration Protocol by providing the job ID.

The server MUST locate the print job by walking every print queue in PrintQueueList, and then walking all print jobs in the **PrintQueue.PrintJobList** of that print queue, until it finds a print job whose PrintJob.ID matches that job ID. If no print job is found, the server returns to the higher-layer application.

If a print job is found, it MUST be removed from PrintQueue.PrintJobList.

3.2.5 Message Processing Events and Sequencing Rules

The server receives the Remote Administration Protocol request from the underlying SMB transport. The server MUST process the request based on the **RAPOpcode** received. The following sections specify the actions the server takes based on the command, as specified by **RAPOpcode**. Once the response is generated, it MUST be sent back to the client.

3.2.5.1 NetShareEnum Command

The Remote Administration Protocol server MUST process NetShareEnumRequest as follows:

The server MUST validate that the incoming **ParamDesc** field of the Remote Administration Protocol request contains the ASCII string "WrLeh"; if it does not, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to ERROR_INVALID_PARAMETER (0x0057), and then return the response to the client.

If the information level is any value other than 0, the server implementing NetShareEnum MUST set the **Win32ErrorCode** value in the Remote Administration Protocol response message to ERROR_INVALID_LEVEL (0x007C).

The Remote Administration Protocol server MUST create a Remote Administration Protocol response message with the **RAPOutParams** set to the contents of a NetShareEnumResponse message. The Remote Administration Protocol server MUST walk the shares in ShareList and fill in the **RAPOutData** field of the Remote Administration Protocol response with as many NetShareInfo structures as can fit within the value specified by the Remote Administration Protocol client's *ReceiveBufferSize* input parameter (see packing rules specified in section 2.5.10) and using the data for the shares, as specified in section 3.2.1.2. The server MUST set the **EntriesReturned** field in NetShareEnumResponse to the number of NetShareInfo structures filled in the **RAPOutData** field of the response.

If the response **EntriesReturned** field is less than the response **EntriesAvailable** field, the NetShareEnum server MUST set the **Win32ErrorCode** value in the Remote Administration Protocol response message to ERROR_MORE_DATA (0x00EA).

If any other errors occur during the response processing, the Remote Administration Protocol server MUST fill the **Win32ErrorCode** value in the Remote Administration Protocol response message with the Win32 error code corresponding to the error. Otherwise, the Remote Administration Protocol server MUST set **Win32ErrorCode** to ERROR_SUCCESS (0X0000).

3.2.5.2 NetServerGetInfo Command

The Remote Administration Protocol server MUST process the NetServerGetInfoRequest as follows:

1. The server MUST validate that the incoming **ParamDesc** field of the Remote Administration Protocol request contains the ASCII string "WrLh"; if it does not, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to ERROR_INVALID_PARAMETER (0x0057), and then return the response to the client.

2. If the information level is any value other than 0 or 1, the server implementing RAP NetServerGetInfo MUST set the Win32ErrorCode value in the RAP response message to ERROR_INVALID_LEVEL (0x007C).

3. The RAP server MUST create an RAP response message with the **RapOutParams** set to the contents of a NetServerGetInfo Response message.

   If the information level of the request was 0, the RAP server MUST walk fill in the **RAPOutData** field of the RAP response with a NetServerInfo0 structure if it can fit within the value specified by the RAP client's ReceiveBufferSize input parameter (see packing rules specified in section 2.5.10). The server MUST set the **TotalBytesAvailable** field in the NetServerGetInfo Response to the number of bytes filled in the **RAPOutData** field of the response.

   If the information level of the request was 1, the RAP server MUST walk fill in the RAPOutData field of the RAP response with a NetServerInfo1 structure if it can fit within the value specified by the RAP client's **ReceiveBufferSize** input parameter (see packing rules specified in section 2.5.10). The server MUST set the **TotalBytesAvailable** field in the NetServerGetInfo Response to the number of bytes filled in the **RAPOutData** field of the response.

   If there is insufficient data to build a response based on the request **ReceiveBufferSize**, the server MUST fail the request with ERROR_INSUFFICIENT_BUFFER (0x007A).

3.2.5.3 NetPrintQEnum Command

The Remote Administration Protocol server MUST process the NetPrintQEnumRequest as follows:

1. The server MUST validate that the incoming **ParamDesc** field of the Remote Administration Protocol request contains the ASCII string "WrLeh"; if it does not, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to ERROR_INVALID_PARAMETER (0x0057), and then return the response to the client.

2. If the information level is any value other than 0x0002, the server implementing NetPrintQEnum MUST set the **Win32ErrorCode** value in the Remote Administration Protocol response message to ERROR_INVALID_LEVEL (0x007C).

   If the input information level is 0x0002, the Remote Administration Protocol server MUST respond to NetPrintQEnumRequest with a Remote Administration Protocol response message with **RAPOutParams** set to the contents of NetPrintQEnumResponse. The Remote Administration Protocol server MUST walk PrintQueueList and fill in the **RAPOutData** field of the Remote Administration Protocol response with as many PrintQueue2 structures as can fit within the value specified by the Remote Administration Protocol client's *ReceiveBufferSize* input parameter (see the packing rules, as specified in section 2.5.10). Immediately following each PrintQueue2 structure, the server MUST walk the PrintQueue.PrintJobList for that queue and fill in as many PrintJobInfo0 structures as is represented in the **PrintJobCount** field in the corresponding PrintQueue structure. If the server cannot fit all of the print jobs within the value specified by the client's *ReceiveBufferSize* parameter, the server MUST remove the PrintQueue2 structure and the related PrintJobInfo0 structures from the buffer, and then return the values (that did fit in the buffer) to the client, as specified below.

3. The server MUST set the **EntriesReturned** field in the response to the number of PrintQueue2 structures in the **RAPOutData** of the response.

   If the response **EntriesReturned** field is less than the response **EntriesAvailable** field, the Remote Administration Protocol NetPrintQEnum server MUST set the **Win32ErrorCode** value in the Remote Administration Protocol response message to ERROR_MORE_DATA (0x00EA).

4. If any other errors occur during the response processing, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response message with the Win32 error code corresponding to the error. Otherwise, the Remote Administration Protocol server MUST set **Win32ErrorCode** to ERROR_SUCCESS (0X0000).

3.2.5.4 NetPrintQGetInfo Command

The Remote Administration Protocol server MUST process the NetPrintQGetInfoRequest as follows:

1. The server MUST validate that the incoming **ParamDesc** field of the Remote Administration Protocol request contains the ASCII string "zWrLh"; if it does not, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to ERROR_INVALID_PARAMETER (0x0057), and then return the response to the client.

2. The Remote Administration Protocol server MUST respond to NetPrintQGetInfoRequest with a Remote Administration Protocol response message with the **RAPOutParams** set to the contents of NetPrintQGetInfoResponse.

3. The Remote Administration Protocol server MUST find the print queue in PrintQueueList by using the name provided in the **PrintQueueName** field of NetPrintQGetInfoRequest. If no print queue is found that matches this name, the server MUST format a Remote Administration Protocol

response with the **Win32ErrorCode** set to NERR_QNotFound, and then return the response to the client.

4. If any other errors occur during the response processing, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response message with the Win32 error code corresponding to the error.

5. If the input information level is 0x0000, the **RAPOutData** data field of the Remote Administration Protocol response MUST be filled with a PrintQueue0 structure, representing the named print queue, using the attributes of the print queue.

   If the input information level is 0x0002, the **RAPOutData** data field of the Remote Administration Protocol response MUST be filled with a PrintQueue2 structure, representing the named print queue, using the attributes of the print queue. Immediately following the PrintQueue2 structure, the server MUST walk PrintQueue.PrintJobList and fill in as many PrintJobInfo0 structures as are represented in the **PrintJobCount** field in the corresponding PrintQueue2 structure.

   If the input information level is 0x0003, the **RAPOutData** field of the Remote Administration Protocol response MUST be filled with PrintQueue3 structures, using the attributes of the print queue. Immediately following the PrintQueue3 structure, the server MUST walk PrintQueue.PrintJobList and fill in as many PrintJobInfo0 structures as are represented in the **PrintJobCount** field in the corresponding PrintQueue3 structure.

6. If the information level is any value other than 0x0002 or 0x0003, the server implementing NetServerEnum MUST set the **Win32ErrorCode** value in the Remote Administration Protocol response message to ERROR_INVALID_LEVEL (0x007C). See sections 2.5.5.2 and 2.5.5.3.

3.2.5.5 NetPrintJobSetInfo Command

The Remote Administration Protocol server MUST process the NetPrintJobSetInfoRequest as follows:

1. The server MUST validate that the incoming **ParamDesc** field of the Remote Administration Protocol request contains the ASCII string "WWsTP"; if it does not, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to ERROR_INVALID_PARAMETER (0x0057), and then return the response to the client.

2. If the information level is any value other than 0x0001, the server implementing Remote Administration Protocol NetServerEnum MUST set the **Win32ErrorCode** value in the Remote Administration Protocol response message to ERROR_INVALID_LEVEL (0x007C).

3. The server MUST locate the print job by walking every print queue in PrintQueueList and by walking all print jobs in the PrintQueue.PrintJobList of that print queue until a job is found whose PrintJob.ID matches what is requested in the NetPrintJobSetInfoRequest. If no print job is found, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to NERR_JobNotFound, and then return the response to the client.

4. The server MUST process the **RAPInData** from the client based on the **ParamNum** field in the NetPrintJobSetInfoRequest. The server MUST look up the **ParamNum** value in the table in section 2.5.7.3.1, and then update the corresponding field in the print job that corresponds to the job ID in the NetPrintJobSetInfoRequest. The server MUST indicate the change to the underlying printer provider, if applicable. The type of the contents of the **RAPInData** field is specified by the Value column in the values given for ParamNum in section 2.5.7.3.1.

5. If any other errors occur during the response processing, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response

message with the Win32 error code corresponding to the error. Otherwise, the Remote Administration Protocol server MUST set **Win32ErrorCode** to ERROR_SUCCESS (0X0000).

3.2.5.6 NetPrintJobGetInfo Command

The Remote Administration Protocol server MUST process the NetPrintJobGetInfoRequest as follows:

The server MUST validate that the incoming **ParamDesc** field of the Remote Administration Protocol request contains the ASCII string "WWrLh"; if it does not, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to ERROR_INVALID_PARAMETER (0x0057), and then return the response to the client.

If the information level of the NetPrintJobGetInfoRequest is any value other than 0x0003, the server implementing Remote Administration Protocol NetServerEnum MUST set the **Win32ErrorCode** value in the Remote Administration Protocol response message to ERROR_INVALID_LEVEL (0x007C).

The server MUST locate the print job by walking every print queue in PrintQueueList and by walking all print jobs in the PrintQueue.PrintJobList of that print queue until a job is found whose PrintJob.ID matches what is requested in the NetPrintJobGetInfoRequest. If no print job is found, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to NERR_JobNotFound, and then return the response to the client.

The **RAPOutData** structure MUST be filled with a PrintJobInfo3 structure, packed as specified in the marshaling rules in section 2.5.10.

If any other errors occur during the response processing, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response message with the Win32 error code corresponding to the error. Otherwise, the Remote Administration Protocol server MUST set **Win32ErrorCode** to ERROR_SUCCESS (0X0000).

3.2.5.7 NetPrintJobDelete Command

The Remote Administration Protocol server MUST process the NetPrintJobDeleteRequest as follows:

The server MUST validate that the incoming **ParamDesc** field of the Remote Administration Protocol request contains the ASCII string "W"; if it does not, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to ERROR_INVALID_PARAMETER (0x0057), and then return the response to the client.

The server MUST locate the print job by walking every print queue in PrintQueueList and by walking all print jobs in the PrintQueue.PrintJobList of that print queue until a job is found whose PrintJob.ID matches what is requested in the NetPrintJobDeleteRequest. If no print job is found, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to NERR_JobNotFound, and then return the response to the client.

The server MUST delete the print job that was found and indicate to the underlying print provider that printing was canceled.

If an error occurs during the response processing, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response message with the Win32 error code corresponding to the error. Otherwise, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response message with ERROR_SUCCESS (0x0000).

3.2.5.8 NetPrintJobPause Command

The Remote Administration Protocol server MUST process the NetPrintJobPauseRequest as follows:

The server MUST validate that the incoming **ParamDesc** field of the Remote Administration Protocol request contains the ASCII string "W"; if it does not, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to ERROR_INVALID_PARAMETER (0x0057), and then return the response to the client.

The server MUST locate the print job by walking every print queue in PrintQueueList and by walking all print jobs in the PrintQueue.PrintJobList of that print queue until a job is found whose PrintJob.ID matches what is requested in the NetPrintJobPauseRequest. If no print job is found, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to NERR_JobNotFound, and then return the response to the client.

The server MUST indicate to the underlying print provider that the print job specified MUST NOT be printed at this time, but MUST remain in the queue.

If an error occurs during the response processing, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response message with the Win32 error code corresponding to the error. Otherwise, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response message with ERROR_SUCCESS (0X0000).

3.2.5.9 NetPrintJobContinue Command

The Remote Administration Protocol server MUST process the NetPrintJobContinueRequest as follows:

The server MUST validate that the incoming **ParamDesc** field of the Remote Administration Protocol request contains the ASCII string "W"; if it does not, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to ERROR_INVALID_PARAMETER (0x0057), and then return the response to the client.

The server MUST locate the print job by walking every print queue in PrintQueueList and by walking all print jobs in the PrintQueue.PrintJobList of that print queue until a job is found whose PrintJob.ID matches what is requested in the NetPrintJobContinueRequest. If no print job is found, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to NERR_JobNotFound, and then return the response to the client.

If the job specified had previously been paused, the server MUST indicate to the underlying print provider that the print job specified MUST be allowed to be printed at this time.

If an error occurs during the response processing, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response message with the Win32 error code corresponding to the error. Otherwise, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response message with ERROR_SUCCESS (0X0000).

3.2.5.10 NetRemoteTOD Command

The Remote Administration Protocol server MUST process the NetRemoteTODRequest as follows:

The server MUST validate that the incoming **ParamDesc** field of the Remote Administration Protocol request contains the ASCII string "rL"; if it does not, the server MUST format a Remote

Administration Protocol response with the **Win32ErrorCode** set to ERROR_INVALID_PARAMETER (0x0057), and then return the response to the client.

The server MUST fill in the **RAPOutData** with a TimeOfDayInfo structure, using the marshaling rules (as specified in section 2.5.10) and the current time.

If any other errors occur during the response processing, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response message with the Win32 error code corresponding to the error. Otherwise, the Remote Administration Protocol server MUST set **Win32ErrorCode** to ERROR_SUCCESS (0X0000).

3.2.5.11 NetServerEnum2 Command

The Remote Administration Protocol server MUST process the NetServerEnum2Request as follows:

The server MUST validate that the incoming **ParamDesc** field of the Remote Administration Protocol request contains the ASCII string "WrLehDz" or "WrLehDO", if it does not, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to ERROR_INVALID_PARAMETER (0x0057), and return the response to the client.

If the information level is any value other than 0 or 1, the server implementing Remote Administration Protocol NetShareEnum MUST set the **Win32ErrorCode** value in the Remote Administration Protocol Response Message to ERROR_INVALID_LEVEL (0x007C).

If the **Flags** field in the incoming NetServerEnum2Request contains the "LL" bit, the server MUST return only those servers (or domains) that exist on the same subnet as the server. If the server cannot determine the list of servers on the current subnet, or its list of servers (or domains) on the current subnet is empty, it MUST return an empty set of servers (or domains).

If the **Flags** field in the incoming NetServerEnum2Request contains the "DL" bit, the server MUST return its list of domains, not its list of servers.

The Remote Administration Protocol server MUST respond to the NetServerEnum2Request with a Remote Administration Protocol Response Message with the **RAPOutParams** set to the contents of a NetServerEnum2Response.

If the InfoLevel of the NetServerEnum2Request structure is 0x0000, the Remote Administration Protocol server MUST walk the ServerList and fill in the **RAPOutData** field of the Remote Administration Protocol response with as many NetServerInfo0 structures based on as many servers in the list as can fit within the value specified by the Remote Administration Protocol client's *ReceiveBufferSize* input parameter (see the packing rules, as specified in section 2.5.10). The server MUST set the **EntriesReturned** field in the NetServerEnum2Response to the number of NetServerInfo0 structures in the **RAPOutData** field of the response.

If the InfoLevel of the NetServerEnum2Request structure is 1, the Remote Administration Protocol server MUST fill in the **RAPOutData** field of the Remote Administration Protocol response with as many NetServerInfo1 structures as can fit within the value specified by the Remote Administration Protocol client's *ReceiveBufferSize* input parameter (see the packing rules, as specified in section 2.5.10). The server MUST set the **EntriesReturned** field in the NetServerEnum2Response to the number of NetServerInfo1 structures in the **RAPOutData** field of the response.

If the response **EntriesReturned** field is less than the response **EntriesAvailable** field, the Remote Administration Protocol server MUST set the **Win32ErrorCode** value in the Remote Administration Protocol response message to ERROR_MORE_DATA (0x00EA).

If any other errors occur during the response processing, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response message

with the Win32 error code corresponding to the error. Otherwise, the Remote Administration Protocol server MUST set **Win32ErrorCode** to ERROR_SUCCESS (0X0000).

3.2.5.12 NetUserPasswordSet2 Command

The Remote Administration Protocol server MUST process the NetUserPasswordSet2Request as follows:

The server MUST validate that the incoming **ParamDesc** field of the Remote Administration Protocol request contains the ASCII string "zb16b16WW"; if it does not, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to ERROR_INVALID_PARAMETER (0x0057), and then return the response to the client.

If the input *EncryptData* parameter is not 0x0000, the Remote Administration Protocol server MUST set the **Win32ErrorCode** value in the Remote Administration Protocol response message to ERROR_INVALID_PARAMETER (0x0057).

The server MUST verify that the old password matches User.Password. If it does not, the server MUST fail the request by creating a Remote Administration Protocol response message with the **Win32ErrorCode** set to ERROR_ACCESS_DENIED. If it does match, the server MUST locate the User in UserList, using the **UserName** field in the NetUserPasswordSet2Request and update its password (or password hash) to the new value.

If any other errors occur during the response processing, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response message with the Win32 error code corresponding to the error. Otherwise, the Remote Administration Protocol server MUST set **Win32ErrorCode** to ERROR_SUCCESS (0X0000).

3.2.5.13 NetServerEnum3 Command

The Remote Administration Protocol server MUST process the NetServerEnum3Request as follows:

The server MUST validate that the incoming **ParamDesc** field of the Remote Administration Protocol request contains the ASCII string "WrLehDzz"; if it does not, the server MUST format a Remote Administration Protocol response with the **Win32ErrorCode** set to ERROR_INVALID_PARAMETER (0x0057), and then return the response to the client.

If the information level is any value other than 0x0000 or 0x0001, the server implementing NetShareEnum MUST set the **Win32ErrorCode** value in the Remote Administration Protocol response message to ERROR_INVALID_LEVEL (0x007c).

If the **Flags** field in the incoming NetServerEnum3Request contains the "LL" bit, the server MUST return only those servers (or domains) that exist on the same subnet as the server. If the server cannot determine the list of servers on the current subnet, or its list of servers (or domains) on the current subnet is empty, it MUST return an empty set of servers (or domains).

If the **Flags** field in the incoming NetServerEnum3Request contains the "DL" bit, the server MUST return its list of domains, not its list of servers.

When determining the set of servers (or domains) to return, the Remote Administration Protocol NetServerEnum3 command processor MUST scan through ServerList, and return entries in its list of servers starting with the entry named in the *FirstNameToReturn* parameter. If the FirstNameToReturn string is empty (a single null character), the browser server SHOULD return entries starting with the first server. If the server's list of servers does not contain the FirstNameToReturn entry, it SHOULD return an empty list of servers.

The Remote Administration Protocol server MUST respond to the NetServerEnum3Request with a Remote Administration Protocol response message with the RAPOutParams set to the contents of a NetServerEnum3Response.

If the InfoLevel of the NetServerEnum3Request structure is 0x0000, the Remote Administration Protocol server MUST walk the entries in ServerList, starting with the first entry to return, as specified above, and fill in the **RAPOutData** field of the Remote Administration Protocol response with as many NetServerInfo0 structures as can fit within the value specified by the Remote Administration Protocol client's *ReceiveBufferSize* input parameter (see the packing rules, as specified in section 2.5.10). The server MUST set the **EntriesReturned** field in the NetServerEnum3Response to the number of NetServerInfo0 structures in the **RAPOutData** field of the response.

If the InfoLevel of the NetServerEnum3Request structure is 0x0001, the Remote Administration Protocol server MUST walk the entries in ServerList, starting with the first entry to return, as specified above, and fill in the **RAPOutData** field of the Remote Administration Protocol response with as many NetServerInfo1 structures as can fit within the value specified by the Remote Administration Protocol client's *ReceiveBufferSize* input parameter (see the packing rules, as specified in section 2.5.10). The server MUST set the **EntriesReturned** field in the NetServerEnum3Response to the number of NetServerInfo1 structures in the **RAPOutData** field of the response.

If the response in the **EntriesReturned** field is less than the response in the **EntriesAvailable** field, the Remote Administration Protocol NetServerEnum3 server MUST set the **Win32ErrorCode** value in the Remote Administration Protocol response message to ERROR_MORE_DATA (0x00EA).

If any other errors occur during the response processing, the Remote Administration Protocol server MUST fill in the **Win32ErrorCode** value in the Remote Administration Protocol response message with the Win32 error code corresponding to the error. Otherwise, the Remote Administration Protocol server MUST set **Win32ErrorCode** to ERROR_SUCCESS (0X0000).

3.2.6 Timer Events

No timer events are required to implement the Remote Administration Protocol server.

3.2.7 Other Local Events

No local events are required to implement the Remote Administration Protocol server.

4 Protocol Examples

4.1 NetShareEnum

The following diagram demonstrates the steps taken to enumerate the shares on a remote server using the Remote Administration Protocol. Assume that this sequence is executed over an existing SMB connection established between the client and the server. The underlying SMB transaction request and response are included for clarity.

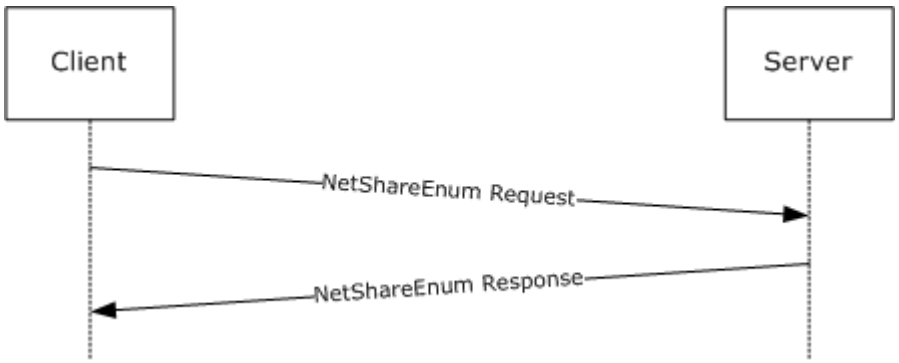


Figure 1: Enumeration of shares

1. The client sends a Remote Administration Protocol request for the NetShareEnum command to the server in an SMB transaction request.

```
Smb: C; Transact, FileName = \PIPE\LANMAN
 Protocol: SMB
 Command: Transact 37(0x25)
 DOSError: No Error
  ErrorClass: No Error
  Reserved: 0 (0x0)
  Error: No Error
 SMBHeader: Command, TID: 0x0800, PID: 0x74B2, UID: 0x0800,
            MID: 0x4681
  Flags: 0 (0x0)
  Flags2: 32768 (0x8000)
  PIDHigh: 0 (0x0)
  SecuritySignature: 0x0
  Reserved: 0 (0x0)
  TreeID: 2048 (0x800)
  ProcessID: 29874 (0x74B2)
  UserID: 2048 (0x800)
  MultiplexID: 18049 (0x4681)
 CTransaction:
  WordCount: 14 (0xE)
  TotalParameterCount: 19 (0x13)
  TotalDataCount: 0 (0x0)
  MaxParameterCount: 8 (0x8)
  MaxDataCount: 4096 (0x1000)
  MaxSetupCount: 0 (0x0)
  Reserved1: 0 (0x0)
  Flags: Do not disconnect TID
   BIT0: ..............0 Do not disconnect TID
  Timeout: 5000 sec(s)
  Reserved2: 0 (0x0)
  ParameterCount: 19 (0x13)
  ParameterOffset: 90 (0x5A)
  DataCount: 0 (0x0)
```

```
 DataOffset: 0 (0x0)
 SetupCount: 0 (0x0)
 Reserved3: 0 (0x0)
 ByteCount: 46 (0x2E)
 Pad: 210 (0xD2)
 UnicodeFileName: \PIPE\LANMAN
 Parameters: RAPParams and NetShareEnum request (19 Bytes)
    00 00 57 72 4C 65 68 00 42 31 33 42 57 7A 00 01   (..WrLeh.B13BWz..)
    00 00 10                                          (...)
```

2. The server responds with the list of shares for this server. In this situation, the server has four shares: C$ with a Remark of "Default share", IPC$ with a Remark of "Remote IPC", ADMIN$ with a Remark of "Remote Admin", and D$ with a Remark of "Default share".

```
Smb: R; Transact
 Protocol: SMB
 Command: Transact 37(0x25)
 DOSError: No Error
  ErrorClass: No Error
  Reserved: 0 (0x0)
  Error: No Error
 SMBHeader: Response, TID: 0x0800, PID: 0x74B2, UID: 0x0800,
            MID: 0x4681
  Flags: 128 (0x80)
  Flags2: 32768 (0x8000)
  PIDHigh: 0 (0x0)
  SecuritySignature: 0x0
  Reserved: 0 (0x0)
  TreeID: 2048 (0x800)
  ProcessID: 29874 (0x74B2)
  UserID: 2048 (0x800)
  MultiplexID: 18049 (0x4681)
 RTransaction:
  WordCount: 10 (0xA)
  TotalParameterCount: 8 (0x8)
  TotalDataCount: 132 (0x84)
  Reserved: 0 (0x0)
  ParameterCount: 8 (0x8)
  ParameterOffset: 56 (0x38)
  ParamDisplacement: 0 (0x0)
  DataCount: 132 (0x84)
  DataOffset: 64 (0x40)
  DataDisplacement: 0 (0x0)
  SetupCount: 0 (0x0)
  Reserved1: 0 (0x0)
  ByteCount: 141 (0x8D)
  Pad1: Binary Large Object (1 Bytes)
  Parameters: ErrorCode, Converter, and RAPOutParams for
              NetShareEnum (8 Bytes)
     00 00 7C 0F 04 00 04 00                           (..|.....)
  Data: RAP NetShareInfo Array (132 Bytes)
     43 24 00 00 00 00 00 00 00 00 00 00 00 00 00 00   (C$..............)
     F2 0F 00 00 49 50 43 24 00 00 00 00 00 00 00 00   (ò...IPC$........)
     00 00 03 00 E7 0F 00 00 41 44 4D 49 4E 24 00 00   (....ç...ADMIN$..)
     00 00 00 00 00 00 00 00 DA 0F 00 00 44 24 00 00   (........Ú...D$..)
     00 00 00 00 00 00 00 00 00 00 00 00 CC 0F 00 00   (............Ì...)
     44 65 66 61 75 6C 74 20 73 68 61 72 65 00 52 65   (Default share.Re)
     6D 6F 74 65 20 41 64 6D 69 6E 00 52 65 6D 6F 74   (mote Admin.Remot)
     65 20 49 50 43 00 44 65 66 61 75 6C 74 20 73 68   (e IPC.Default sh)
     61 72 65 00                                       (are.)
```

4.2 NetServerEnum2

The following diagram demonstrates the steps taken to retrieve an enumeration of servers on the network from a remote server using the Remote Administration Protocol. Assume that this sequence is executed over an existing SMB connection established between the client and the server. The underlying SMB transaction request and response are included for clarity.

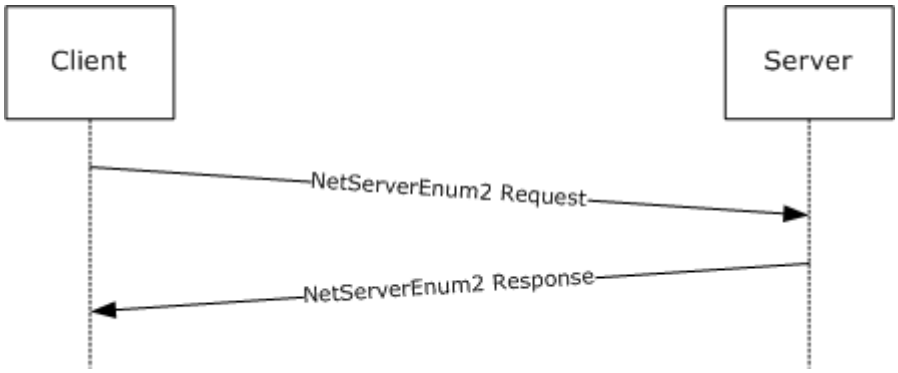


Figure 2: Enumeration of servers

1. The client sends a Remote Administration Protocol request for the NetServerEnum2 command to the server in an SMB transaction request.

```
Smb: C; Transact, FileName = \PIPE\LANMAN
 Protocol: SMB
 Command: Transact 37(0x25)
 DOSError: No Error
  ErrorClass: No Error
  Reserved: 0 (0x0)
  Error: No Error
 SMBHeader: Command, TID: 0x0801, PID: 0x74B2, UID: 0x0802,
            MID: 0x1B02
  Flags: 0 (0x0)
  Flags2: 32768 (0x8000)
  PIDHigh: 0 (0x0)
  SecuritySignature: 0x0
  Reserved: 0 (0x0)
  TreeID: 2049 (0x801)
  ProcessID: 29874 (0x74B2)
  UserID: 2050 (0x802)
  MultiplexID: 6914 (0x1B02)
 CTransaction:
  WordCount: 14 (0xE)
  TotalParameterCount: 26 (0x1A)
  TotalDataCount: 0 (0x0)
  MaxParameterCount: 8 (0x8)
  MaxDataCount: 6144 (0x1800)
  MaxSetupCount: 0 (0x0)
  Reserved1: 0 (0x0)
  Flags: Do not disconnect TID
   BIT0: ...............0 Do not disconnect TID
  Timeout: 5000 sec(s)
  Reserved2: 0 (0x0)
  ParameterCount: 26 (0x1A)
  ParameterOffset: 90 (0x5A)
  DataCount: 0 (0x0)
  DataOffset: 0 (0x0)
  SetupCount: 0 (0x0)
  Reserved3: 0 (0x0)
```

```
 ByteCount: 53 (0x35)
 Pad: 113 (0x71)
 UnicodeFileName: \PIPE\LANMAN
 Parameters: RAPParams and NetServerEnum2 Request (26 Bytes)
    68 00 57 72 4C 65 68 44 4F 00 42 31 36 42 42 44   (h.WrLehDO.B16BBD)
    7A 00 01 00 00 18 FF FF FF FF                      (z.....ÿÿÿÿ)
```

2. The server responds with the list of servers on the network. In this case, there are 12 servers to be returned, and all 12 are returned in this response.

```
Smb: R; Transact
 Protocol: SMB
 Command: Transact 37(0x25)
 DOSError: No Error
 ErrorClass: No Error
  Reserved: 0 (0x0)
  Error: No Error
 SMBHeader: Response, TID: 0x0801, PID: 0x74B2, UID: 0x0802,
            MID: 0x1B02
  Flags: 128 (0x80)
  Flags2: 32768 (0x8000)
  PIDHigh: 0 (0x0)
  SecuritySignature: 0x0
  Reserved: 0 (0x0)
  TreeID: 2049 (0x801)
  ProcessID: 29874 (0x74B2)
  UserID: 2050 (0x802)
  MultiplexID: 6914 (0x1B02)
 RTransaction:
  WordCount: 10 (0xA)
  TotalParameterCount: 8 (0x8)
  TotalDataCount: 379 (0x17B)
  Reserved: 0 (0x0)
  ParameterCount: 8 (0x8)
  ParameterOffset: 56 (0x38)
  ParamDisplacement: 0 (0x0)
  DataCount: 379 (0x17B)
  DataOffset: 64 (0x40)
  DataDisplacement: 0 (0x0)
  SetupCount: 0 (0x0)
  Reserved1: 0 (0x0)
  ByteCount: 388 (0x184)
  Pad1: Binary Large Object (1 Bytes)
 Parameters: ErrorCode, Converter, and RAPOutParams for
             NetServerEnum2 Response (8 Bytes)
    00 00 85 16 0B 00 0B 00                            (........)
 Data: RAP NetServerInfo1 Array (379 Bytes)
    42 52 55 43 43 4F 2D 4F 46 46 33 00 00 00 00 00   (BRUCCO-OFF3.....)
    05 02 03 92 82 00 FF 17 00 00 53 4D 42 4E 54 34   (...??.ÿ...SMBNT4)
    53 52 56 00 00 00 00 00 00 00 04 00 03 90 01 00   (SRV...........□..)
    FE 17 00 00 53 4D 42 57 46 57 33 31 31 00 00 00   (þ...SMBWFW311...)
    00 00 00 00 01 33 03 20 01 00 CD 17 00 00 53 4D   (.....3. ..Í...SM)
    42 57 49 4E 32 30 30 30 00 00 00 00 00 00 05 00   (BWIN2000........)
    03 90 02 02 CC 17 00 00 53 4D 42 57 49 4E 32 30   (.□..Ì...SMBWIN20)
    30 33 00 00 00 00 00 00 05 02 03 90 82 00 CB 17   (03.........□?.Ë.)
    00 00 53 4D 42 57 49 4E 32 30 30 33 49 41 36 34   (..SMBWIN2003IA64)
    00 00 05 02 03 90 82 00 CA 17 00 00 53 4D 42 57   (.....□?.Ê...SMBW)
    49 4E 39 38 53 45 00 00 00 00 00 00 04 00 03 20   (IN98SE......... )
    41 00 B8 17 00 00 53 4D 42 57 49 4E 39 38 53 45   (A.¸...SMBWIN98SE)
    2D 55 4D 00 00 00 00 04 00 03 20 41 00 A6 17 00 00 (-UM..... A.¦...)
    53 4D 42 57 49 4E 58 50 00 00 00 00 00 00 00 00   (SMBWINXP........)
    05 01 03 10 00 00 A5 17 00 00 53 50 53 4D 42 44   (......¥...SPSMBD)
    43 31 00 00 00 00 00 00 00 00 05 00 03 90 82 02   (C1...........□?.)
```

```
A4 17 00 00 53 50 53 4D 42 44 43 32 00 00 00 00   (¤...SPSMBDC2....)
00 00 00 00 05 02 2B 10 84 00 A3 17 00 00 00 00   (......+.?.£.....)
00 57 49 4E 53 45 20 46 49 4C 45 20 53 59 53 54   (.WINSE FILE SYST)
45 4D 00 57 49 4E 53 45 20 46 49 4C 45 20 53 59   (EM.WINSE FILE SY)
53 54 45 4D 00 00 00 00 31 32 33 34 35 36 37 38   (STEM....12345678)
39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34   (9012345678901234)
35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30   (5678901234567890)
31 32 33 34 35 36 37 38 00 00 00                  (12345678...)
```

4.3 NetPrintJobDel

The following diagram demonstrates the steps taken to enumerate the deletion of a print job on a remote server by using the Remote Administration Protocol. Assume that this sequence is executed over an existing SMB connection established between the client and the server, and that the identifier of the job being deleted is 3. The underlying SMB transaction request and response are included for clarity.

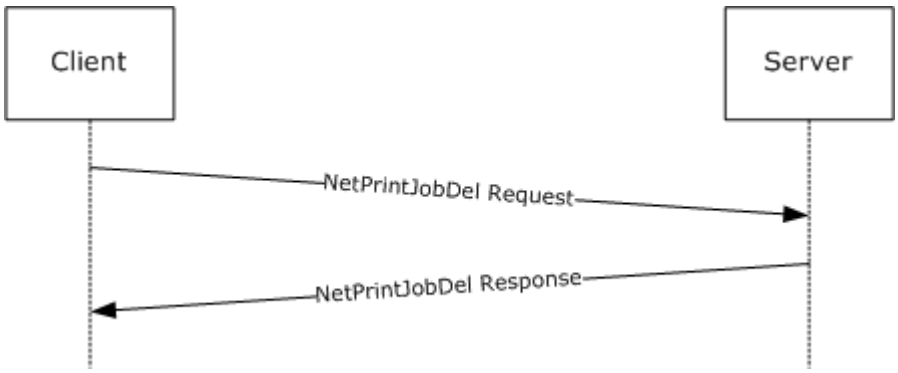


Figure 3: Deletion of a print job

1. The client sends a Remote Administration Protocol request for the NetPrintJobDel command to the server in an SMB transaction request.

```
Smb: C; Transact, FileName = \PIPE\LANMAN
 Protocol: SMB
 Command: Transact 37(0x25)
 DOSError: No Error
  ErrorClass: No Error
  Reserved: 0 (0x0)
  Error: No Error
 SMBHeader: Command, TID: 0x0802, PID: 0x74B2, UID: 0x0801,
            MID: 0x6D81
  Flags: 0 (0x0)
  Flags2: 32768 (0x8000)
  PIDHigh: 0 (0x0)
  SecuritySignature: 0x0
  Reserved: 0 (0x0)
  TreeID: 2050 (0x802)
  ProcessID: 29874 (0x74B2)
  UserID: 2049 (0x801)
  MultiplexID: 28033 (0x6D81)
 CTransaction:
```

```
 WordCount: 14 (0xE)
 TotalParameterCount: 7 (0x7)
 TotalDataCount: 0 (0x0)
 MaxParameterCount: 4 (0x4)
 MaxDataCount: 0 (0x0)
 MaxSetupCount: 0 (0x0)
 Reserved1: 0 (0x0)
 Flags: Do not disconnect TID
  BIT0: ..............0 Do not disconnect TID
 Timeout: 5000 sec(s)
 Reserved2: 0 (0x0)
 ParameterCount: 7 (0x7)
 ParameterOffset: 90 (0x5A)
 DataCount: 0 (0x0)
 DataOffset: 0 (0x0)
 SetupCount: 0 (0x0)
 Reserved3: 0 (0x0)
 ByteCount: 34 (0x22)
 Pad: 83 (0x53)
 UnicodeFileName: \PIPE\LANMAN
 Parameters: RAPParams and NetPrintJobDel Request (7 Bytes)
    51 00 57 00 00 03 00                              (Q.W....)
```

2. The server deletes the print job and returns success.

```
Smb: R; Transact
 Protocol: SMB
 Command: Transact 37(0x25)
 DOSError: No Error
  ErrorClass: No Error
  Reserved: 0 (0x0)
  Error: No Error
 SMBHeader: Response, TID: 0x0802, PID: 0x74B2, UID: 0x0801,
            MID: 0x6D81
  Flags: 128 (0x80)
  Flags2: 32768 (0x8000)
  PIDHigh: 0 (0x0)
  SecuritySignature: 0x0
  Reserved: 0 (0x0)
  TreeID: 2050 (0x802)
  ProcessID: 29874 (0x74B2)
  UserID: 2049 (0x801)
  MultiplexID: 28033 (0x6D81)
 RTransaction:
  WordCount: 10 (0xA)
  TotalParameterCount: 4 (0x4)
  TotalDataCount: 0 (0x0)
  Reserved: 0 (0x0)
  ParameterCount: 4 (0x4)
  ParameterOffset: 56 (0x38)
  ParamDisplacement: 0 (0x0)
  DataCount: 0 (0x0)
  DataOffset: 60 (0x3C)
  DataDisplacement: 0 (0x0)
```

```
SetupCount: 0 (0x0)
Reserved1: 0 (0x0)
ByteCount: 5 (0x5)
Pad1: Binary Large Object (1 Bytes)
Parameters: RAPOutParams (4 Bytes)
   00 00 00 00                                    (....)
```

5 Security

5.1 Security Considerations for Implementers

The Remote Administration Protocol uses descriptor strings to define the data being passed between the client and the server. As such, an implementer might implement a generic parsing engine that can parse the data from the Remote Administration Protocol client (using the data provided by the Remote Administration Protocol client) without validation.

If a server implements such an engine, it opens the possibility for buffer overruns and other attacks caused by a client that passes parameter descriptors and data descriptors that do not match the expected values. Care must be taken when implementing the Remote Administration Protocol to ensure that Remote Administration Protocol servers do not trust the values of the request data passed by the client, and to ensure that Remote Administration Protocol clients do not trust the values of the responses from the server.

In addition, several of the Remote Administration Protocol request and response structures contain **Pad** fields, which are normally ignored. A server or client that does not set the contents of the **Pad** fields to a known value (such as null) runs the risk of enabling an information disclosure attack against the server or client.

Password operations specified for the Remote Administration Protocol send the password in clear text over the network, and thus are not secure. This should be considered before using them to change passwords.

5.2 Index of Security Parameters

None.

6 Appendix A: Windows Behavior

The information in this specification is applicable to versions of Windows as follows:

- Windows NT
- Windows 2000
- Windows XP
- Windows Server 2003
- Windows Vista

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 1.6: This protocol is supported on all versions of the Windows operating system, including Windows 2000 Server, Windows XP, Windows Server 2003, and Windows Vista, for purposes of compatibility with pre-Windows NT clients. With the exception of the NetServerEnum2 command and the NetServerEnum3 command, the only client that uses this protocol is Windows 98. The Remote Administration Protocol is obsolete and is used primarily for communications with Windows 98 clients. The Windows 98 operating system uses the Remote Administration Protocol for the following operations:

- File Share enumeration (NET VIEW \\SERVER)
- Printer management
- Time retrieval
- Server list retrieval

In addition, the CIFS Browser Protocol uses the Remote Administration Protocol to retrieve lists of servers.

<2> Section 1.7: As shipped, Windows 98 clients use only the NTLM Authentication Protocol. If the Active Directory Client Extension is installed on the Windows 98 machine, the Windows 98 machine uses the NTLM Authentication Protocol.

<3> Section 1.8: Windows uses only the values specified in [MS-ERREF] section 3.

<4> Section 2.5.1: The Windows SMB server validates that the **ParamDesc** matches what is specified by the **RAPOpcode**; if they do not match, the server returns ERROR_INVALID_PARAMETER (0x57), as specified in section 2.5.2.

<5> Section 2.5.1: The current Windows implementation of the Remote Administration Protocol ignores the **DataDesc** field.

<6> Section 2.5.1: Previous implementations of Windows used the supplied **DataDesc**, **ParamDesc**, and **AuxDesc** structures to convert the RAPParams and the response structures to and from "C" style structures. Current versions of Windows simply validate that the **ParamDesc** supplied by the client is the value required by the **RAPOpcode**.

<7> Section 2.5.5.4.2: The following table shows the unsigned 8-bit major operating system version number thatWindows clients and servers use:

Value	Meaning
0x04	Operating system is Windows 95, Windows 98, Windows Me, or Windows NT 4.0.
0x05	Operating system is Windows 2000 Server, Windows XP, Windows Server 2003, or Windows Server 2003 R2.
0x06	Operating system is Windows Server 2008 or Windows Vista.

<8> Section 2.5.5.4.2: The following table shows the unsigned 8-bit minor operating system version number that Windows clients and servers use:

Value	Meaning
0x00	Operating system is Windows 95Windows NT 4.0, Windows 2000 Server, Windows Server 2008, or Windows Vista.
0x01	Operating system is Windows XP.
0x02	Operating system is Windows XP Professional x64 Edition, Windows Server 2003, or Windows Server 2003 R2.
0x0A	Operating system is Windows 98.
0x5A	Operating system is Windows Me.

<9> Section 2.5.8.1.1:

- The **RealPasswordLength** is used only for password length restriction checks.
- The password fields are not encrypted, and the **EncryptedPassword** field is always set to 0.
- The contents of the **OldPassword** and **NewPassword** fields (past the end of the **OldPassword** and **NewPassword** fields) are not initialized and MUST be ignored.

7 Index

O

P

R

S

T

U

V

W