[MS-FAX]:
Fax Server and Client Remote 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	1.2.2	Editorial	Revised and edited the technical content.
07/20/2007	1.2.3	Editorial	Revised and edited the technical content.
08/10/2007	1.3	Minor	Updated the technical content.
09/28/2007	1.3.1	Editorial	Revised and edited the technical content.
10/23/2007	1.4	Minor	Updated references.
11/30/2007	1.4.1	Editorial	Revised and edited the technical content.
01/25/2008	1.4.2	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction

The Fax Server and Client Remote Protocol Specification defines a Microsoft-proprietary protocol that is referred to as the FAX Server and Client Remote Protocol. This protocol is a **remote procedure call (RPC)**–based, client/server protocol based on that is used to send faxes and manage the fax server and its **queues**.

1.1 Glossary

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

Authentication Level
Authentication Service (AS)
Dynamic Endpoint
Endpoint
Globally Unique Identifier (GUID)
Handle
Interface Definition Language (IDL)
Microsoft Interface Definition Language (MIDL)
Network Data Representation (NDR)
Opnum
Remote Procedure Call (RPC)
RPC Protocol Sequence
RPC Transport
Security Provider
Universal Unique Identifier (UUID)
Well-Known Endpoint

The following terms are specific to this document:

Activity Logging: A log provided by the fax service that can log incoming and outgoing fax activity such as **job** identifiers, submission time, banner contents, status, call time, file name, and other fax-specific information. This activity logging is configurable by the fax server administrator.

Archive: The folder where successful faxes are stored. There is an outgoing **archive** (called Sent Items in the **Fax Console**), where successfully sent faxes are stored, and an incoming **archive** (called Inbox in the **Fax Console**), where successfully received faxes are stored.

Body: The fax pages other than the cover page.

Broadcast: An action of sending the same fax to multiple **recipients**.

Calling Subscriber ID (CSID): A text string that identifies a fax **recipient**. The Calling Subscriber ID (CSID) is transmitted to a fax sender by the receiving device when an incoming fax is detected. The **CSID** is often a combination of the fax number and business name. It is often the same as the transmitting station identifier (**TSID**).

Coordinated Universal Time (UTC): A universal timekeeping standard that is based on Greenwich Mean Time (GMT). Local time is calculated in **UTC** and offset by the local time zone.

CSID: See **Calling Subscriber ID**.

Device (or Port): A fax **device** that is used by the fax service to send or receive faxes.

Document: A fax that has not yet been submitted to a fax server. A **document** can consist of a cover page and **body**, but must include at least a cover page or body.

Fax Body: See **Body**.

Fax Console: The fax service user interface that is used to manage incoming and outgoing faxes.

Fax Document: See **Document**.

Fax Job: See **Job**.

Fax Message: See **Message**.

Fax Routing Extension: A DLL that is used by the fax service and that exposes one or more inbound routing methods. These methods are used by the fax service to automatically route incoming faxes.

Fax Routing Method: A routing method that is used by the fax service to automatically route incoming faxes.

Fax Service Provider (FSP): A DLL that is used by the fax service and that exposes one or more fax **devices** to the service. The DLL coordinates between the fax service and the fax **device**.

General Configuration: A set of properties on the fax server that defines the overall fax service behavior. These properties include the number of retries that should be attempted while sending a fax, the delay between each retry, the number of days unsent **jobs** are retained, branding, and application of telephone discount rates. These properties are configurable.

Job: An inbound or outbound fax transmission that is in a fax server's **queue**.

Message: A fax that a fax server has successfully received or transmitted and **archived**.

Outbound Rule: A rule that specifies whether a fax is sent by using either a specific **device** or a group of **devices**. If the telephone number for an outgoing fax matches the area code and country/region code of a routing rule, the fax service sends the fax according to the matching routing rule.

Outbound Groups: A group that specifies the routing group by which the fax service sends a fax for which the routing rule applies. A routing group must be created before it is specified in a routing rule.

Queue: The folder where faxes that are being processed (**jobs**) are stored. There is an outgoing **queue** (called Outbox in the **Fax Console**) where faxes that are being sent are stored. There also is an incoming **queue** (called Incoming in the **Fax Console**) where faxes that are being received are stored.

Recipient: The **recipient** of a **fax message**.

SMTP: The Simple Mail Transfer Protocol, as defined in [RFC821].

Tagged Image File Format: See **TIFF**.

TIFF: A high-resolution, tag-based image format. **Tagged Image File Format** (**TIFF**) is used for the universal interchange of digital images.

Transmitting Station ID (TSID): A string that specifies the transmitter subscriber ID that is sent by the fax machine when it sends a fax to a receiving machine. This string is usually a combination of the fax or telephone number and the name of the business. It is often the same as the **Calling Subscriber ID**.

TSID: See **Transmitting Station ID**.

UTC: See **Coordinated Universal Time**.

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

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

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

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

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

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

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

[MS-SMB] Microsoft Corporation, "Server Message Block (SMB) Protocol Specification", July 2007.

[RFC821] Postel, J., "Simple Mail Transfer Protocol", RFC 821, August 1982, http://www.ietf.org/rfc/rfc0821.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

[MSDN-AS] Microsoft Corporation, "Authorization Structures", http://msdn2.microsoft.com/en-us/library/aa375780.aspx

[MSDN-AUTHN] Microsoft Corporation, "Authentication-Service Constants", http://msdn2.microsoft.com/en-us/library/aa373556.aspx

[MSDN-FAX_LOG_CATEGORY] Microsoft Corporation, "FAX_LOG_CATEGORY Structure", http://msdn2.microsoft.com/en-us/library/ms690890.aspx

[MSDN-FRM] Microsoft Corporation, "Fax Routing Methods", http://msdn2.microsoft.com/en-us/library/ms691955.aspx

[MSDN-FSCAR] Microsoft Corporation, "Fax Service Client API Reference", http://msdn2.microsoft.com/en-us/library/ms692335.aspx

[MSDN-SAR] Microsoft Corporation, "Standard Access Rights", http://msdn2.microsoft.com/en-us/library/aa379607.aspx

[MSDN-SECURITY_INFORMATION] Microsoft Corporation, "SECURITY_INFORMATION", http://msdn2.microsoft.com/en-us/library/aa379573.aspx

[RFC3302] Parsons, G. and Rafferty, J., "Tag Image File Format (TIFF) - image/tiff MIME Sub-Type Registration", RFC 3302, September 2002, http://www.ietf.org/rfc/rfc3302.txt

1.3 Protocol Overview (Synopsis)

The FAX Server and Client Remote Protocol manages and sends faxes, manages the fax server and its queues, and allows fax clients to act as RPC servers so that they can accept status notifications from fax servers acting as clients.

1.3.1 Fax Server Protocol

The FAX Server and Client Remote Protocol can be used to submit and manage faxes. It can be further used to change configuration on the fax server; for example, setting the Inbound Routing Rules/**Outbound Groups**. The FAX Server and Client Remote Protocol can be used to change settings—such as whether the fax service should **archive** the faxes it sends or receives, the number of days the fax service should keep an archive, or the number of rings before a call should be answered. Practically everything that manages the behavior of the fax server can be controlled by using the FAX Server and Client Remote Protocol.

This can be used either locally, where both the client and server are on the same machine, or remotely, where the client and server are on different machines.

Fax server provides for custom **Fax Service Providers (FSPs)** via the following RPC calls:

- FAX_RegisterServiceProviderEx
- FAX_UnRegisterServiceProviderEx

For these RPC calls, the vendor needs to register/unregister by using a **GUID**.

1.3.2 Fax Client Protocol

The FAX Server and Client Remote Protocol is used for notifications. When activity occurs on the server—for example, when a new fax is received, a change occurs in the status of an outgoing fax, or a change occurs in configuration—events are generated. Clients that want to register for these events can act like RPC servers with the fax server as the RPC client and get these events with the event type and event data as described in section 3.2.

This can be used either locally, where both the client and server are on the same machine, or remotely, where the client and server are on different machines.

1.4 Relationship to Other Protocols

The FAX Server and Client Remote Protocol is dependent on the following protocols:

- RPC

- TCP/IP (for RPC over TCP/IP)
- Named pipes

No protocols are dependent on the FAX Server and Client Remote Protocol.

1.5 Prerequisites/Preconditions

This protocol defines RPC interfaces, and therefore has the prerequisites specified in [MS-RPCE] section 1.5, as being common to RPC interfaces.

It is assumed that the protocol client has obtained the name of a server that supports the FAX Server and Client Remote Protocol before this protocol is invoked.

1.6 Applicability Statement

The FAX Server and Client Remote Protocol is applicable only for operations between a computer that functions as a client and a computer that functions as a fax server. The protocol is intended for communicating status, setting configuration, and submitting **jobs** and notification data between fax server and client applications.

The protocol can be used in a broad set of scenarios ranging from a home-use scenario, where one computer makes its fax server available for use by other computers, to an enterprise-use scenario where a fax server provides faxing services for many computers.

1.7 Versioning and Capability Negotiation

- Supported Transports: This protocol uses RPC over TCP only.
- Protocol Versions: This protocol has only one interface version, but that interface has been extended by adding additional methods at the end and the version number has been changed for newer clients to use the newer methods. Few methods have been replaced by newer methods. The use of these methods is specified in 3.1.
- A server in a domain uses the default server principal name for the Simple and Protected GSSAPI Negotiation (SPNEGO) **security provider**, the authentication-service constant RPC_C_AUTHN_LEVEL_PKT_PRIVACY. For general information concerning Windows authentication-service constants, see [MSDN-AUTHN].
- An RPC client uses the default server principal name for the SPNEGO security provider, the authentication-service constant RPC_C_AUTHN_LEVEL_PKT_PRIVACY. An RPC client always uses the packet **authentication level**, as specified in [MS-RPCE] section 3.3.1.5.2.
- Localization: The protocol does not contain locale-specific information.
- Capability Negotiation: No capability negotiation mechanism is built into the protocol.

1.8 Vendor-Extensible Fields

There are no vendor-extensible fields.

1.9 Standards Assignments

Parameter	Value	Reference
RPC **UUID** for FaxClient	6099fc12-3eff-11d0-abd0-	[C706],

Parameter	Value	Reference
	00c04fd91a4e	Appendix A
RPC UUID for SHAREDFAX interface	ea0a3165-4834-11d2-a6f8-00c04fa346cc	[C706], Appendix A
String for named pipe for **well-known endpoint** for local connections	SHAREDFAX	Section 2.1
String for named pipe for well-known endpoint for server connections	\\<server machine name>\\SHAREDFAX	Section 2.1
Pipe name	\\PIPE\SHAREDFAX	Section 2.1

2 Messages

The following sections specify how **messages** are transported and supply details of message syntax, including common structures, certificate requirements, and common error codes.

2.1 Transport

The FAX Server and Client Remote Protocol uses the transport RPC over TCP/IP, as specified in [MS-RPCE] section 2.1.1.1.

This protocol uses RPC well-known endpoints. This is a named pipe that has the value server machine name followed by SHAREDFAX for remote and SHAREDFAX for local.

This protocol uses RPC **dynamic endpoints**, as specified in [C706].

This protocol MUST use the UUIDs as specified in section 1.9.

2.2 Common Data Types

This protocol MUST indicate to the RPC runtime that it is to support the NDR20 transfer syntax only, as specified in [C706] Part 4.

This protocol MUST enable the ms_union extension as specified in [MS-RPCE] section 2.2.4.

In addition to RPC base types and definitions specified in [C706] and [MS-DTYP], additional data types are defined in the following sections.

2.2.1 FAX_ENUM_MESSAGE_FOLDER

The **FAX_ENUM_MESSAGE_FOLDER** enumeration defines possible locations for a **fax message**.

```
typedef  enum
{
  FAX_MESSAGE_FOLDER_INBOX = 0x00000000,
  FAX_MESSAGE_FOLDER_SENTITEMS = 0x00000001,
  FAX_MESSAGE_FOLDER_QUEUE = 0x00000002
} FAX_ENUM_MESSAGE_FOLDER;
```

FAX_MESSAGE_FOLDER_INBOX: The incoming fax transmissions archive.

FAX_MESSAGE_FOLDER_SENTITEMS: The outgoing fax transmissions archive.

FAX_MESSAGE_FOLDER_QUEUE: The outgoing and incoming fax queue.

2.2.2 FAX_ENUM_CONFIG_OPTION

The **FAX_ENUM_CONFIG_OPTION** enumeration identifies the configuration option to be returned by the **FAX_GetConfigOption** method.

```
typedef  enum
{
  FAX_CONFIG_OPTION_ALLOW_PERSONAL_CP = 0x00000000,
  FAX_CONFIG_OPTION_QUEUE_STATE = 0x00000001,
```

```
  FAX_CONFIG_OPTION_ALLOWED_RECEIPTS = 0x00000002,
  FAX_CONFIG_OPTION_INCOMING_FAXES_PUBLIC = 0x00000003
} FAX_ENUM_CONFIG_OPTION;
```

FAX_CONFIG_OPTION_ALLOW_PERSONAL_CP: Represents whether the server allows personal cover pages or not.

FAX_CONFIG_OPTION_QUEUE_STATE: Corresponds to the state of the queue.

FAX_CONFIG_OPTION_ALLOWED_RECEIPTS: Corresponds to the type of receipts that the server is configured to send.

FAX_CONFIG_OPTION_INCOMING_FAXES_PUBLIC: Corresponds to the viewing permissions of incoming faxes.

2.2.3 FAX_ENUM_PERSONAL_PROF_TYPES

The **FAX_ENUM_PERSONAL_PROF_TYPES** enumeration defines values to indicate personal profile types.

```
typedef  enum
{
  RECIPIENT_PERSONAL_PROF = 0x00000001,
  SENDER_PERSONAL_PROF = 0x00000002
} FAX_ENUM_PERSONAL_PROF_TYPES;
```

RECIPIENT_PERSONAL_PROF: Indicates a **recipient** profile.

SENDER_PERSONAL_PROF: Indicates a sender profile.

2.2.4 FAX_JOB_ENTRY

The **FAX_JOB_ENTRY** structure describes one **fax job**. The structure includes data on the job type and status, recipient and sender identification, scheduling and delivery settings, and the page count. The **SizeOfStruct** and **RecipientNumber** members MUST NOT be 0.

A fax client application passes the **FAX_JOB_ENTRY** structure in a call to the **FAX_SetJob** function.

An application can call the **FAX_EnumJobs** function to enumerate all queued and active fax jobs on the fax server of interest. **FAX_EnumJobs** returns an array of **FAX_JOB_ENTRY** structures. Each structure describes one fax job in detail.

```
typedef struct FAX_JOB_ENTRY {
  DWORD SizeOfStruct;
  DWORD JobId;
  LPCTSTR UserName;
  DWORD JobType;
  DWORD QueueStatus;
  DWORD Status;
  DWORD Size;
  DWORD PageCount;
  LPCTSTR RecipientNumber;
  LPCTSTR RecipientName;
```

```
} FAX_JOB_ENTRY,
 *PFAX_JOB_ENTRY;
```

SizeOfStruct: A DWORD that MUST specify the size, in bytes, of the **FAX_JOB_ENTRY** structure. The calling application MUST set this member to sizeof(**FAX_JOB_ENTRY**) before it calls the **FAX_SetJob** function.

JobId: A DWORD that MUST specify a unique number that identifies the fax jobs of interest. This number must match the value that the calling application passes in the *JobId* parameter to the **FAX_SetJob** function.

UserName: A pointer to a constant null-terminated character string that MUST specify the name of the user who submitted the fax jobs.

JobType: A DWORD variable that MUST specify the type of the fax jobs of interest. This member MUST be one of the following job types.

Value	Meaning
JT_UNKNOWN 0x0001	The job type is unknown. This value indicates that the fax server has not yet scheduled the job.
JT_SEND 0x0002	The job is an outgoing fax transmission.
JT_RECEIVE 0x0004	The job is an incoming fax transmission.
JT_ROUTING 0x0008	The fax server tried to route the fax transmission, but routing failed. The fax server will attempt to route the job again.
JT_FAIL_RECEIVE 0x0010	The fax server did not route the fax because it did not receive the entire transmission. The fax server saves the partial transmission in a temporary directory.

QueueStatus: A DWORD variable that MUST be set of bit flags indicating the queue status of the fax jobs identified by the **JobId** member. This member SHOULD be one or more of the following values.

Value	Meaning
JS_PENDING 0x00000001	The fax jobs is in the queue and pending service.
JS_INPROGRESS 0x00000002	The fax jobs is in progress.
JS_DELETING 0x00000004	The fax server is deleting the fax jobs.
JS_FAILED 0x00000008	The fax jobs failed.
JS_PAUSED 0x00000010	The fax server paused the fax jobs.

Value	Meaning
JS_NOLINE 0x00000020	There is no line available to send the fax. The fax server will send the transmission when a line is available.
JS_RETRYING 0x00000040	The fax jobs failed. The fax server will attempt to retransmit the fax after a specified interval.
JS_RETRIES_EXCEEDED 0x00000080	The fax server exceeded the maximum number of retransmission attempts allowed. The fax will not be sent.

Status: A DWORD variable that MUST specify a fax **device** status code or value. This value MUST be one of the following predefined device status codes.

Value	Meaning
FPS_DIALING 0x20000001	The device is dialing a fax number.
FPS_SENDING 0x20000002	The device is sending a **fax document**.
FPS_RECEIVING 0x20000004	The device is receiving a fax document.
FPS_COMPLETED 0x20000008	The device completed sending or receiving a fax transmission.
FPS_HANDLED 0x20000010	The fax service processed the outbound fax document; the fax service provider will transmit the **document**.
FPS_UNAVAILABLE 0x20000020	The device is not available because it is in use by another application.
FPS_BUSY 0x20000040	The device encountered a busy signal.
FPS_NO_ANSWER 0x20000080	The receiving device did not answer the call.
FPS_BAD_ADDRESS 0x20000100	The device dialed an invalid fax number.
FPS_NO_DIAL_TONE 0x20000200	The sending device cannot complete the call because it does not detect a dial tone.
FPS_DISCONNECTED 0x20000400	The fax call was disconnected by the sender or the caller.
FPS_FATAL_ERROR 0x20000800	The device has encountered a fatal protocol error.
FPS_NOT_FAX_CALL 0x20001000	The device received a call that was a data call or a voice call.
FPS_CALL_DELAYED 0x20002000	The device delayed a fax call because the sending device received a busy signal multiple times. The device cannot retry the call because dialing restrictions exist (some countries/regions restrict the number of retry attempts when a number is busy).

Value	Meaning
FPS_CALL_BLACKLISTED 0x20004000	The device could not complete a call because the telephone number was blocked or reserved; emergency numbers such as 911 are blocked.
FPS_INITIALIZING 0x20008000	The device is initializing a call.
FPS_OFFLINE 0x20010000	The device is offline and unavailable.
FPS_RINGING 0x20020000	The device is ringing.
FPS_AVAILABLE 0x20100000	The device is available.
FPS_ABORTING 0x20200000	The device is aborting a fax job.
FPS_ROUTING 0x20400000	The device is routing a received fax document.
FPS_ANSWERED 0x20800000	The device answered a new call.

Size: A DWORD variable that MUST specify the size, in bytes, of the fax document to transmit. The size MUST not exceed 4 gigabytes (GB).

PageCount: A DWORD variable that MUST specify the total number of pages in the fax transmission.

RecipientNumber: A pointer to a constant null-terminated character string that MUST specify the fax number of the recipient of the fax transmission.

RecipientName: A pointer to a constant null-terminated character string that MUST specify the name of the recipient of the fax.

2.2.5 FAX_PORT_INFO

The **FAX_PORT_INFO** structure describes one fax port. The data includes, among other items, a device identifier, the port's name and current status, and station identifiers.

A fax client application passes the **FAX_PORT_INFO** structure in a call to the **FAX_SetPort** function to modify the configuration of the fax port of interest.

If an application calls the **FAX_EnumPorts** function to enumerate all the fax devices currently attached to a fax server, the function returns an array of **FAX_PORT_INFO** structures. Each structure describes one device in detail. If an application calls the **FAX_GetPort** function to query one device, that function returns information about the device in one **FAX_PORT_INFO** structure.

```
typedef struct FAX_PORT_INFO {
  DWORD SizeOfStruct;
  DWORD DeviceId;
  DWORD State;
  DWORD Flags;
  DWORD Rings;
```

```
  DWORD Priority;
  LPCTSTR DeviceName;
  LPCTSTR Tsid;
  LPCTSTR Csid;
} FAX_PORT_INFO,
 *PFAX_PORT_INFO;
```

SizeOfStruct: A DWORD that MUST specify the size, in bytes, of the FAX_PORT_INFO structure. The calling application SHOULD ensure that this member is set to sizeof(**FAX_PORT_INFO**) before it calls the **FAX_SetPort** function.

DeviceId: A DWORD variable that MUST specify the permanent line identifier for the fax device of interest.

State: A DWORD variable that MUST specify a fax device status code or value. This member MUST be one of the following predefined device status codes:

Value	Meaning
FPS_DIALING 0x20000001	The device is dialing a fax number.
FPS_SENDING 0x20000002	The device is sending a fax document.
FPS_RECEIVING 0x20000004	The device is receiving a fax document.
FPS_COMPLETED 0x20000008	The device completed sending or receiving a fax transmission.
FPS_HANDLED 0x20000010	The fax service processed the outbound fax document; the fax service provider will transmit the document.
FPS_UNAVAILABLE 0x20000020	The device is not available because it is in use by another application.
FPS_BUSY 0x20000040	The device encountered a busy signal.
FPS_NO_ANSWER 0x20000080	The receiving device did not answer the call.
FPS_BAD_ADDRESS 0x20000100	The device dialed an invalid fax number.
FPS_NO_DIAL_TONE 0x20000200	The sending device cannot complete the call because it does not detect a dial tone.
FPS_DISCONNECTED 0x20000400	The fax call was disconnected by the sender or the caller.
FPS_FATAL_ERROR 0x20000800	The device has encountered a fatal protocol error.
FPS_NOT_FAX_CALL	The device received a call that was a data call or a voice call.

Value	Meaning
0x20001000	
FPS_CALL_DELAYED 0x20002000	The device delayed a fax call because the sending device received a busy signal multiple times. The device cannot retry the call because dialing restrictions exist (some countries/regions restrict the number of retry attempts when a number is busy).
FPS_CALL_BLACKLISTED 0x20004000	The device could not complete a call because the telephone number was blocked or reserved; emergency numbers such as 911 are blocked.
FPS_INITIALIZING 0x20008000	The device is initializing a call.
FPS_OFFLINE 0x20010000	The device is offline and unavailable.
FPS_RINGING 0x20020000	The device is ringing.
FPS_AVAILABLE 0x20100000	The device is available.
FPS_ABORTING 0x20200000	The device is aborting a fax job .
FPS_ROUTING 0x20400000	The device is routing a received fax document.
FPS_ANSWERED 0x20800000	The device answered a new call.

Flags: A DWORD variable that MUST specify a set of bit flags that specify the capability of the fax port. This member MAY be a combination of the following values:

Value	Meaning
FPF_RECEIVE 0x00000001	The device can receive faxes.
FPF_SEND 0x00000002	The device can send faxes.
FPF_VIRTUAL 0x00000004	The device is a virtual fax device. Note that you cannot set a device to be virtual. When calling **FAX_GetPort**, the FAX_PORT_INFO flag's FPF_VIRTUAL value indicates whether the device is virtual. When calling **FAX_SetPort**, the service will only relate to the FPF_RECEIVE and FPF_SEND values.

Rings: A DWORD variable that MUST indicate the number of times an incoming fax call should ring before the specified device answers the call. Values MUST be from 0 to 99 inclusive. This value SHOULD be ignored unless the FPF_RECEIVE port capability bit flag is set.

Priority: A DWORD variable that SHOULD determine the relative order in which available fax devices send outgoing transmissions. Values for this member MUST be 1 through n, where n is the value of the *PortsReturned* parameter returned by a call to the **FAX_EnumPorts** function. When the fax server initiates an outgoing fax transmission, it attempts to select the device with the highest priority and FPF_SEND port capability. If that device is not available, the

server SHOULD select the next available device that follows in rank order, and so on. The value of the **Priority** member has no effect on incoming transmissions.

DeviceName: A pointer to a constant null-terminated character string that MUST specify the name of the fax device of interest.

Tsid: A pointer to a constant null-terminated character string that MUST specify the **transmitting station identifier** (**TSID**). This identifier is usually a telephone number. Only printable characters such as English letters, numeric symbols, and punctuation marks (ASCII range 0x20 to 0x7F) MUST be used in a TSID.

Csid: A pointer to a constant null-terminated character string that MUST specify the called station identifier (**CSID**). This identifier is usually a telephone number. Only printable characters, such as English letters, numeric symbols, and punctuation marks (ASCII range 0x20 to 0x7F) MUST be used in a CSID.

2.2.6 FAX_ROUTING_METHOD

The **FAX_ROUTING_METHOD** structure contains information about one **fax routing method**, as it pertains to one fax device. The data includes, among other items, whether the fax routing method is enabled for the device and the name of the binary that exports the routing method. It also includes the GUID and function name that uniquely identify the routing method and the method's user-friendly name.

A fax client application can call the **FAX_EnumRoutingMethods** function to enumerate all the fax routing methods associated with a specific fax device. The function returns an array of **FAX_ROUTING_METHOD** structures. Each structure describes one fax routing method in detail.

Call the **FAX_EnableRoutingMethod** function to enable or disable a fax routing method for a specific fax device.

```
typedef struct FAX_ROUTING_METHOD {
  DWORD SizeOfStruct;
  DWORD DeviceId;
  BOOL Enabled;
  LPCTSTR DeviceName;
  LPCTSTR Guid;
  LPCTSTR FriendlyName;
  LPCTSTR FunctionName;
  LPCTSTR ExtensionImageName;
  LPCTSTR ExtensionFriendlyName;
} FAX_ROUTING_METHOD,
 *PFAX_ROUTING_METHOD;
```

SizeOfStruct: A DWORD that MUST specify the size, in bytes, of the **FAX_ROUTING_METHOD** structure. The fax service sets this member to sizeof(**FAX_ROUTING_METHOD** (section 2.2.6)).

DeviceId: A DWORD variable that MUST specify the permanent line identifier for the fax device of interest.

Enabled: A Boolean variable that MUST specify whether the fax routing method is enabled or disabled for the fax device of interest. If this parameter is equal to TRUE, the fax routing method is enabled for the device.

DeviceName: A pointer to a constant null-terminated character string that MUST specify the name of the fax device of interest.

Guid: A pointer to a constant null-terminated character string that MUST specify the GUID that uniquely identifies the fax routing method of interest.

FriendlyName: A pointer to a constant null-terminated character string that MUST specify the user-friendly name to display for the fax routing method.

FunctionName: A pointer to a null-terminated character string that MUST specify the name of the function that executes the specified fax routing procedure. The **fax routing extension** binary identified by the **ExtensionImageName** member exports the function.

ExtensionImageName: A pointer to a constant null-terminated character string that MUST specify the name of the fax routing extension binary that implements the fax routing method.

ExtensionFriendlyName: A pointer to a constant null-terminated character string that MUST specify the user-friendly name to display for the fax routing extension binary.

2.2.7 FAX_DEVICE_STATUS

The **FAX_DEVICE_STATUS** structure contains information about the current status of a fax device. In addition to the status, the structure also includes data on whether the device is currently sending or receiving a fax transmission, device and station identifiers, sender and recipient names, and routing information.

The fax client application can call the **FAX_GetDeviceStatus** function to retrieve status information for the fax device of interest. The function returns the information in a **FAX_DEVICE_STATUS** structure.

```
typedef struct FAX_DEVICE_STATUS {
  DWORD SizeOfStruct;
  LPCTSTR CallerId;
  LPCTSTR Csid;
  DWORD CurrentPage;
  DWORD DeviceId;
  LPCTSTR DeviceName;
  LPCTSTR DocumentName;
  DWORD JobType;
  LPCTSTR PhoneNumber;
  LPCTSTR RoutingString;
  LPCTSTR SenderName;
  LPCTSTR RecipientName;
  DWORD Size;
  FILETIME StartTime;
  DWORD Status;
  LPCTSTR StatusString;
  FILETIME SubmittedTime;
  DWORD TotalPages;
  LPCTSTR Tsid;
  LPCTSTR UserName;
} FAX_DEVICE_STATUS,
 *PFAX_DEVICE_STATUS;
```

SizeOfStruct: A DWORD that MUST specify the size, in bytes, of the **FAX_DEVICE_STATUS** structure. The fax service sets this member to sizeof(**FAX_DEVICE_STATUS**).

CallerId: If the **JobType** member is equal to the JT_RECEIVE job type, CallerId SHOULD be a pointer to a null-terminated character string that identifies the calling device that sent the active fax document. This string MAY include the telephone number of the calling device.

Csid: A pointer to a constant, null-terminated character string that MUST specify the called station identifier of the device.

CurrentPage: A DWORD that MUST specify the number of the page in the fax transmission that the fax device is currently sending or receiving. The page count MUST be relative to one.

DeviceId: A DWORD that MUST specify the permanent line identifier for the fax device of interest.

DeviceName: A pointer to a constant, null-terminated character string that MUST specify the name of the fax device of interest.

DocumentName: A pointer to a constant, null-terminated character string that MUST specify the document name to associate with the fax document that the device is currently sending or receiving. This is the user-friendly name that appears in the print spooler.

JobType: A DWORD that MUST specify the type of fax job that is currently active on the device. This member MUST be one of the following job types.

Value	Meaning
JT_UNKNOWN 0x0001	The fax device is in an unknown or idle state.
JT_SEND 0x0002	The fax device is sending a fax document.
JT_RECEIVE 0x0004	The fax device is receiving a fax document.

PhoneNumber: If the **JobType** member is equal to the JT_SEND job type, PhoneNumber SHOULD be a pointer to a constant null-terminated character string that is the fax number dialed for the outgoing fax transmission.

RoutingString: If the **JobType** member is equal to the JT_RECEIVE job type, RoutingString SHOULD be a pointer to a constant null-terminated character string that specifies the routing string for an incoming fax. The string MUST be of the form:

Canonical-Phone-Number[|Additional-Routing-Info]

where:

Canonical-Phone-Number is defined in the Address topic of the Telephony Application Programming Interface (TAPI) documentation (see the Canonical Address subheading); and

Additional-Routing-Info is the subaddress of a canonical address and uses the subaddress format.

SenderName: A pointer to a constant, null-terminated character string that MUST specify the name of the sender who initiated the fax transmission.

RecipientName: A pointer to a constant, null-terminated character string that MUST specify the name of the recipient of the fax transmission.

Size: A DWORD that MUST specify the size, in bytes, of the active fax document.

StartTime: A FILETIME structure that MUST specify the starting time of the current fax job. The time MUST be expressed in **coordinated universal time** (**UTC**).

Status: A DWORD variable that MUST specify a fax device status code or value. This member MUST be one of the following predefined device status codes:

Value	Meaning
FPS_DIALING 0x20000001	The device is dialing a fax number.
FPS_SENDING 0x20000002	The device is sending a fax document.
FPS_RECEIVING 0x20000004	The device is receiving a fax document.
FPS_COMPLETED 0x20000008	The device completed sending or receiving a fax transmission.
FPS_HANDLED 0x20000010	The fax service processed the outbound fax document; the fax service provider will transmit the document.
FPS_UNAVAILABLE 0x20000020	The device is not available because it is in use by another application.
FPS_BUSY 0x20000040	The device encountered a busy signal.
FPS_NO_ANSWER 0x20000080	The receiving device did not answer the call.
FPS_BAD_ADDRESS 0x20000100	The device dialed an invalid fax number.
FPS_NO_DIAL_TONE 0x20000200	The sending device cannot complete the call because it does not detect a dial tone.
FPS_DISCONNECTED 0x20000400	The fax call was disconnected by the sender or the caller.
FPS_FATAL_ERROR 0x20000800	The device has encountered a fatal protocol error.
FPS_NOT_FAX_CALL 0x20001000	The device received a call that was a data call or a voice call.
FPS_CALL_DELAYED 0x20002000	The device delayed a fax call because the sending device received a busy signal multiple times. The device cannot retry the call because dialing restrictions exist (some countries/regions restrict the number of retry attempts when a number is busy).
FPS_CALL_BLACKLISTED 0x20004000	The device could not complete a call because the telephone number was blocked or reserved; emergency numbers such as 911 are blocked.

Value	Meaning
FPS_INITIALIZING 0x20008000	The device is initializing a call.
FPS_OFFLINE 0x20010000	The device is offline and unavailable.
FPS_RINGING 0x20020000	The device is ringing.
FPS_AVAILABLE 0x20100000	The device is available.
FPS_ABORTING 0x20200000	The device is aborting a fax job.
FPS_ROUTING 0x20400000	The device is routing a received fax document.
FPS_ANSWERED 0x20800000	The device answered a new call.

StatusString: This member MUST be NULL.

SubmittedTime: Specifies a **FILETIME** structure that MUST contain the time the client submitted the fax document for transmission to the fax job queue. The time MUST be expressed in UTC.

TotalPages: A DWORD that MUST specify the total number of pages in the fax transmission.

Tsid: A pointer to a constant, null-terminated character string that MUST specify the transmitting station identifier (TSID). This identifier is usually a telephone number.

UserName: A pointer to a constant null-terminated character string that MUST specify the name of the user who submitted the active fax job.

2.2.8 FAX_LOG_CATEGORY

The **FAX_LOG_CATEGORY** structure describes one logging category. The structure contains a logging category name and identifier. It also includes the current level at which the fax server logs events for the specified logging category in the application event log. The fax client application passes the **FAX_LOG_CATEGORY** structure in a call to the **FAX_SetLoggingCategories** function to modify the current logging categories for the fax server of interest. If the application calls the **FAX_GetLoggingCategories** function, it returns the current settings in a **FAX_LOG_CATEGORY**.

```
typedef struct FAX_LOG_CATEGORY {
  LPCTSTR Name;
  DWORD Category;
  DWORD Level;
} FAX_LOG_CATEGORY,
 *PFAX_LOG_CATEGORY;
```

Name: A pointer to a null-terminated character string that MUST specify a descriptive name for the logging category.

Category: A DWORD that MUST specify a unique value that identifies a logging category. This member MUST be one of the following predefined values.

Value	Meaning
FAXLOG_CATEGORY_INIT 1	A fax service initialization or termination event.
FAXLOG_CATEGORY_OUTBOUND 2	An outgoing fax transmission event such as sending a fax.
FAXLOG_CATEGORY_INBOUND 3	An incoming fax transmission event such as receiving a fax or routing a fax.
FAXLOG_CATEGORY_UNKNOWN 4	An unknown event.

Level: A DWORD that MUST specify the current logging level for the logging category. This member MUST be one of the following predefined logging levels.

Value	Meaning
FAXLOG_LEVEL_NONE 0	The fax server MUST not log events.
FAXLOG_LEVEL_MIN 1	The fax server SHOULD log only the most severe failure events.
FAXLOG_LEVEL_MED 2	The fax server SHOULD log most events (this level does not include some informational and warning events).
FAXLOG_LEVEL_MAX 3	The fax server MUST log all events.

2.2.9 FAX_COVERPAGE_INFO_EXW

The **FAX_COVERPAGE_INFO_EXW** structure defines members that specify information about the fax cover page.

```
typedef struct FAX_COVERPAGE_INFO_EX {
  DWORD dwSizeOfStruct;
  DWORD dwCoverPageFormat;
  LPWSTR lptstrCoverPageFileName;
  BOOL bServerBased;
  LPWSTR lptstrNote;
  LPWSTR lptstrSubject;
} FAX_COVERPAGE_INFO_EXW,
 *PFAX_COVERPAGE_INFO_EXW,
 *LPCFAX_COVERPAGE_INFO_EXW;
```

dwSizeOfStruct: A DWORD that MUST specify the size of this structure.

dwCoverPageFormat: A DWORD that MUST indicate the format of the cover page template. This MUST be one of the values defined in **FAX_ENUM_COVERPAGE_FORMATS**.

lptstrCoverPageFileName: A pointer to a null-terminated character string that MUST specify the fully qualified path to the cover page template file.

bServerBased: A Boolean that must indicate whether the cover page template specified in the *lptstrCoverPageFileName* parameter is a personal cover page or a server-based cover page. TRUE, if the *lptstrCoverPageFileName* parameter specifies a server-based page without any path designations. The server SHOULD use the \templates directory to find the cover page.

lptstrNote: A pointer to a null-terminated string that MUST specify the content for the **note** field of the cover page.

lptstrSubject: A pointer to a null-terminated string that MUST specify the content for the **subject** field.

2.2.10 FAX_JOB_PARAM_EXW

The **FAX_JOB_PARAM_EXW** structure defines information about the new job to create when sending a fax message.

```
typedef struct FAX_JOB_PARAM_EXW {
  DWORD dwSizeOfStruct;
  DWORD dwScheduleAction;
  SYSTEMTIME tmSchedule;
  DWORD dwReceiptDeliveryType;
  LPWSTR lptstrReceiptDeliveryAddress;
  FAX_ENUM_PRIORITY_TYPE Priority;
  HCALL hCall;
  DWORD_PTR dwReserved[4];
  LPWSTR lptstrDocumentName;
  DWORD dwPageCount;
} FAX_JOB_PARAM_EXW,
 *PFAX_JOB_PARAM_EXW,
 *LPCFAX_JOB_PARAM_EXW;
```

dwSizeOfStruct: A DWORD that MUST contain the size of this structure. The value SHOULD be the size of the FAX_JOB_PARAM_EXW structure.

dwScheduleAction: A DWORD that MUST specify when to send the fax. This member MUST be one of the following values:

Value	Meaning
JSA_NOW 0	Send the fax as soon as a device is available.
JSA_SPECIFIC_TIME 1	Send the fax at the time specified by the **tmSchedule** member.
JSA_DISCOUNT_PERIOD 2	Send the fax during the discount rate period. Call the **FAX_GetConfiguration** function to retrieve the discount period for the fax server.

tmSchedule: A **SYSTEMTIME** structure that MUST contain the date and time to send the fax. The time MUST be specified In Coordinated Universal Time (UTC). This parameter SHOULD be

ignored unless **dwScheduleAction** is set to 1 (JSA_SPECIFIC_TIME). If the time specified has already passed, the method behaves as if 0 (JSA_NOW) was specified.

dwReceiptDeliveryType: A DWORD that MUST specify the type of receipt delivered to the sender when the fax is successfully sent and when the fax transmission fails. It MAY also specify if a receipt will be sent for each recipient or for all the recipients together. The value of this parameter MUST be a logical combination of one of the delivery method flags and any of the delivery grouping flags as specified in **FAX_ENUM_DELIVERY_REPORT_TYPES**.

lptstrReceiptDeliveryAddress: A pointer to a constant, null-terminated character string. If the **dwReceiptDeliveryType** member is equal to DRT_EMAIL, the string SHOULD be the address to which the Delivery Receipt (DR) or Non-Delivery Receipt (NDR) should be sent. If the **dwReceiptDeliveryType** member is equal to DRT_INBOX, the string SHOULD be the MAPI profile to which the DR or **NDR** should be sent. If the **dwReceiptDeliveryType** member is equal to DRT_MSGBOX, the string SHOULD be the computer name upon which the DR or NDR message box should appear.

Priority: A value specifying the priority level of the outgoing fax.

hCall: Reserved.

Note This value MUST be set to NULL.

dwReserved: SHOULD be set to zero.

lptstrDocumentName: Pointer to a constant, null-terminated character string to associate with the fax document. This is the user-friendly name that appears in the print spooler.

dwPageCount: A DWORD value that MUST specify the number of pages in the document pointed to by the *lpctstrFileName* parameter of the **FAX_SendDocumentEx** method. This value MUST be used only for **Tagged Image File Format (TIFF)**documents, which is the only supported format.

2.2.11 FAX_MESSAGE_PROPS

The **FAX_MESSAGE_PROPS** structure defines the properties of a fax message that can be set.

```
typedef struct FAX_MESSAGE_PROPS {
  DWORD dwValidityMask;
  DWORD dwMsgFlags;
} FAX_MESSAGE_PROPS,
 *PFAX_MESSAGE_PROPS;
```

dwValidityMask: A DWORD value that MUST define a bitwise combination of valid fields in the structure.

Value	Meaning
FAX_MSG_PROP_FIELD_MSG_FLAGS 0x0001	Indicates whether the value in dwMsgFlags is valid. If this bit is set, the value in dwMsgFlags is valid.

dwMsgFlags: A DWORD bitmask that MUST indicate the state to which the message flags should be set.

Value	Meaning
FAX_MSG_FLAG_READ 0x00000001	Determines whether this fax message should be marked as read. If this bit is set, the message is marked as read. If this bit is reset, the message is marked as unread.

2.2.12 FAX_OUTBOX_CONFIG

The **FAX_OUTBOX_CONFIG** structure defines information about outbox settings of the fax server.

```
typedef struct _FAX_OUTBOX_CONFIG {
  DWORD dwSizeOfStruct;
  BOOL bAllowPersonalCP;
  BOOL bUseDeviceTSID;
  DWORD dwRetries;
  DWORD dwRetryDelay;
  FAX_TIME dtDiscountStart;
  FAX_TIME dtDiscountEnd;
  DWORD dwAgeLimit;
  BOOL bBranding;
} FAX_OUTBOX_CONFIG,
 *PFAX_OUTBOX_CONFIG;
```

dwSizeOfStruct: A DWORD that MUST specify the size of the structure.

bAllowPersonalCP: A Boolean that MUST indicate whether fax client applications can include a user-designed cover page with the fax transmission or not. If this member is TRUE, the client MAY provide a personal cover page file. If this member is FALSE, the client MUST use a common cover page stored on the fax server.

bUseDeviceTSID: A Boolean variable that MUST indicate whether the fax server MAY use the devices transmitting station identifier instead of the value specified when submitting a new job. If this member is TRUE, the server SHOULD use the devices transmitting station identifier.

dwRetries: A DWORD that MUST specify the number of times the fax server will attempt to retransmit an outgoing fax if the initial transmission fails.

dwRetryDelay: A DWORD that MUST specify the minimum number of minutes that will elapse between retransmission attempts by the fax server.

dtDiscountStart: A **FAX_TIME** structure that MUST indicate the hour and minute at which the discount period begins. The discount period applies only to outgoing transmissions.

dtDiscountEnd: A **FAX_TIME** structure that MUST indicate the hour and minute at which the discount period ends. The discount period applies only to outgoing transmissions.

dwAgeLimit: A DWORD variable that MUST specify the number of days the fax server will keep unsuccessful fax messages in its outbox queue. If a fax message stays in the outbox queue longer than the value specified, it MAY be automatically deleted. If this value is zero, the time limit MUST not be used.

bBranding: A Boolean that SHOULD specify whether the fax server should generate a brand (banner) at the top of outgoing fax transmissions or not. If this member is TRUE, the fax

server SHOULD generate a brand that contains transmission-related information such as the transmitting station identifier, date, time, and page count.

2.2.13 FAX_REASSIGN_INFO

The **FAX_REASSIGN_INFO** structure contains information about the reassignment of a fax.

```
typedef struct FAX_REASSIGN_INFO {
  LPCWSTR lpcwstrRecipients;
  LPCWSTR lpcwstrSenderName;
  LPCWSTR lpcwstrSenderFaxNumber;
  LPCWSTR lpcwstrSubject;
  BOOL bHasCoverPage;
} FAX_REASSIGN_INFO,
 *PFAX_REASSIGN_INFO;
```

lpcwstrRecipients: A pointer to a constant, null-terminated character string that MUST specify an array of intended recipients to assign the fax message to. The recipients are separated by a semicolon.

lpcwstrSenderName: A pointer to a constant null-terminated character string that MUST specify the sender name for the received fax.

lpcwstrSenderFaxNumber: A pointer to a constant null-terminated character string that MUST specify the sender fax number for the received fax.

lpcwstrSubject: A pointer to a constant, null-terminated character string that MUST specify the subject of the received fax.

bHasCoverPage: Boolean value that MUST indicate whether the fax includes a cover page or not. If this member is TRUE, the fax SHOULD include a cover page; otherwise, it is FALSE.

2.2.14 FAX_RECIPIENT_INFO

The **FAX_RECIPIENT_INFO** structure contains a recipient's personal profile and usage information.

```
typedef struct FAX_RECIPIENT_INFO {
  DWORD dwSizeofStruct;
  PFAX_PERSONAL_PROFILE pRecipientProfile;
  DWORD usage;
} FAX_RECIPIENT_INFO,
 *PFAX_RECIPIENT_INFO;
```

dwSizeofStruct: A DWORD value that MUST specify the size, in bytes, of the structure.

pRecipientProfile: A pointer to a **FAX_PERSONAL_PROFILEW** structure that MUST specify the recipient's personal profile.

usage: A DWORD that MUST specify information about the usage of the message in relation to this recipient. This member MUST contain one of the following values:

Value	Meaning
RECIPIENT_USAGE_TO 0x00000001	Fax is addressed to the recipient.
RECIPIENT_USAGE_CC 0x00000002	Fax is carbon-copied (CC) to the recipient.
RECIPIENT_USAGE_BCC 0x00000003	Fax is blind carbon-copied (BCC) to the recipient.

2.2.15 FAX_SERVER_ACTIVITY

The **FAX_SERVER_ACTIVITY** structure defines information about the fax server queue activity and the events reported by the fax server.

```
typedef struct FAX_SERVER_ACTIVITY {
  DWORD dwSizeOfStruct;
  DWORD dwIncomingMessages;
  DWORD dwRoutingMessages;
  DWORD dwOutgoingMessages;
  DWORD dwDelegatedOutgoingMessage;
  DWORD dwQueuedMessages;
  DWORD dwErrorEvents;
  DWORD dwWarningEvents;
  DWORD dwInformationEvents;
} FAX_SERVER_ACTIVITY,
 *PFAX_SERVER_ACTIVITY;
```

dwSizeOfStruct: Represents the size of the server activity structure.

Note MUST be set to specify the size of the **FAX_SERVER_ACTIVITY** structure.

dwIncomingMessages: A DWORD that indicates the number of message currently being received by the fax server. This variable MAY also be set to the count of the number of incoming messages that were successfully received and are currently being routed using an inbound routing method. If the routing fails, the incoming job SHOULD be marked for a routing retry and the **dwRoutingMessages** member used to count this job when the routing restarts. If this value is nonzero, stopping the server MAY result in the loss of incoming messages.

dwRoutingMessages: A DWORD that indicates the number of incoming messages being rerouted after a routing failure.

dwOutgoingMessages: A DWORD that indicates the number of messages currently being sent by the fax server. If this value is nonzero, stopping the server MAY result in the loss of outgoing messages.

dwDelegatedOutgoingMessage: A DWORD that indicates the number of messages currently being sent by a Fax Service Provider on behalf of the fax server. The fax server is not currently sending these messages.

dwQueuedMessages: A DWORD that indicates the number of outgoing messages waiting to be processed in the fax server's queue.

dwErrorEvents: A DWORD that indicates the number of error entries added to the system event log since the last time the fax server was started.

dwWarningEvents: A DWORD that indicates the number of warning entries added to the system event log since the last time the fax server was started.

dwInformationEvents: A DWORD that indicates the number of information entries added to the system event log since the last time the fax server was started.

2.2.16 FAX_SPECIFIC_ACCESS_RIGHTS

The **FAX_SPECIFIC_ACCESS_RIGHTS** enumeration defines specific access rights, which provide security when users query and manage fax jobs, fax devices, and fax documents.

```
typedef  enum
{
  FAX_ACCESS_SUBMIT = 0x0001,
  FAX_ACCESS_SUBMIT_NORMAL = 0x0002,
  FAX_ACCESS_SUBMIT_HIGH = 0x0004,
  FAX_ACCESS_QUERY_JOBS = 0x0008,
  FAX_ACCESS_MANAGE_JOBS = 0x0010,
  FAX_ACCESS_QUERY_CONFIG = 0x0020,
  FAX_ACCESS_MANAGE_CONFIG = 0x00040,
  FAX_ACCESS_QUERY_IN_ARCHIVE = 0x00080,
  FAX_ACCESS_MANAGE_IN_ARCHIVE = 0x0100,
  FAX_ACCESS_QUERY_OUT_ARCHIVE = 0x0200,
  FAX_ACCESS_MANAGE_OUT_ARCHIVE = 0x0400,
  FAX_GENERIC_READ = FAX_ACCESS_QUERY_JOBS | FAX_ACCESS_QUERY_CONFIG |
FAX_ACCESS_QUERY_IN_ARCHIVE | FAX_ACCESS_QUERY_OUT_ARCHIVE,
  FAX_GENERIC_WRITE = FAX_ACCESS_MANAGE_JOBS | FAX_ACCESS_MANAGE_CONFIG |
FAX_ACCESS_MANAGE_IN_ARCHIVE | FAX_ACCESS_MANAGE_OUT_ARCHIVE,
  FAX_GENERIC_EXECUTE = FAX_ACCESS_SUBMIT,
  FAX_GENERIC_ALL = FAX_ACCESS_SUBMIT |
FAX_ACCESS_SUBMIT_NORMAL | FAX_ACCESS_SUBMIT_HIGH | FAX_ACCESS_QUERY_JOBS |
FAX_ACCESS_MANAGE_JOBS | FAX_ACCESS_QUERY_CONFIG | FAX_ACCESS_MANAGE_CONFIG |
FAX_ACCESS_QUERY_IN_ARCHIVE | FAX_ACCESS_MANAGE_IN_ARCHIVE | FAX_ACCESS_QUERY_OUT_ARCHIVE
| FAX_ACCESS_MANAGE_OUT_ARCHIVE

} FAX_SPECIFIC_ACCESS_RIGHTS;
```

FAX_ACCESS_SUBMIT: The user MAY submit low priority fax messages. The user MAY view and manage messages in the server's queue and outgoing archive.

FAX_ACCESS_SUBMIT_NORMAL: The user MAY submit normal priority fax messages. The user MAY view and manage messages in the server's queue and outgoing archive.

FAX_ACCESS_SUBMIT_HIGH: The user MAY submit high priority fax messages. The user MAY view and manage messages in the server's queue and outgoing archive.

FAX_ACCESS_QUERY_JOBS: The user MAY view all the jobs (incoming/outgoing) in the server's queue.

FAX_ACCESS_MANAGE_JOBS: The user MAY view and manage all the jobs (incoming/outgoing) in the server's queue.

FAX_ACCESS_QUERY_CONFIG: The user MAY view the fax server's configuration.

FAX_ACCESS_MANAGE_CONFIG: The user MAY view and set the fax server's configuration.

FAX_ACCESS_QUERY_IN_ARCHIVE: The user MAY view all messages in the incoming messages archive.

FAX_ACCESS_MANAGE_IN_ARCHIVE: The user MAY view and manage all messages in the incoming messages archive.

FAX_ACCESS_QUERY_OUT_ARCHIVE: The user MAY view all messages in the outgoing messages archive.

FAX_ACCESS_MANAGE_OUT_ARCHIVE: The user MAY view and manage all messages in the outgoing messages archive.

FAX_GENERIC_READ: Access rights needed to read faxes.

FAX_GENERIC_WRITE: Access rights needed to write faxes.

FAX_GENERIC_EXECUTE: Access rights needed to execute faxes.

FAX_GENERIC_ALL: All access rights.

2.2.17 FAX_VERSION

The **FAX_VERSION** structure contains information about the version of the fax server components.

```
typedef struct FAX_VERSION {
  DWORD dwSizeOfStruct;
  BOOL bValid;
  unsigned short wMajorVersion;
  unsigned short wMinorVersio;
  unsigned short wMajorBuildNumber;
  unsigned short wMinorBuildNumber;
  DWORD dwFlags;
} FAX_VERSION,
 *PFAX_VERSION;
```

dwSizeOfStruct: A DWORD that MUST specify the size of the structure.

bValid: A Boolean value indicating the validity of the version information returned.

Note This value MUST be set to false if no version information is returned in this structure.

wMajorVersion: A WORD containing the major version number of the fax server component.

wMinorVersio: A WORD containing the minor version number of the fax server component.

wMajorBuildNumber: A WORD containing the major build number of the fax server component.

wMinorBuildNumber: A WORD containing the minor build number of the fax server component.

dwFlags: A DWORD that MUST specify one of the following values:

Value	Meaning
0x00000000	Indicates that the server component was built in release mode. **Note** If built in release mode, this value MUST be zero, which is the default.
FAX_VER_FLAG_CHECKED 0x00000001	Indicates that the server component was built in debug mode.
FAX_VER_FLAG_EVALUATION 0x00000002	Indicates that the server component was built for evaluation purposes. Reserved for future use.

The following are the custom-marshaled data types that are used in RPC. Most of the custom-marshaled data is used for out parameters in RPC calls and corresponds to already documented FAX data types.

In the following diagrams, only representations on 32-bit operating systems are shown. For 64-bit operating systems, each of the 4-byte pointer fields need to be replaced by an 8-byte pointer field.

String fields contain the offset to the location where the actual string data is stored. The receiver MUST retrieve the string contents from the mentioned offset.

LPWSTR, used in the following, is defined as follows:

```
typedef wchar_t * LPWSTR
```

Similarly, LPCWSTR is defined as follows:

```
typedef const wchar_t * LPCWSTR
```

2.2.18 FAX_ACCOUNT_INFO_0

An array of the **FAX_ACCOUNT_INFO_0** data type MAY be passed as an out parameter (as a byte array) in a **FAX_EnumAccounts (Opnum 95)**() call. This data type MAY also be passed as an out parameter (as a byte array) in a **FAX_GetAccountInfo (Opnum 96)**() call. Because this array is passed as a byte array, it requires custom marshaling. For reference, the data type definition is shown below:

```
typedef struct FAX_ACCOUNT_INFO_0 {
  DWORD dwSizeOfStruct;
  LPCWSTR lpcwstrAccountName;
} FAX_ACCOUNT_INFO_0,
 *PFAX_ACCOUNT_INFO_0;
```

dwSizeOfStruct: A DWORD value that MUST indicate the size, in bytes, of the structure.

lpcwstrAccountName: A pointer to a constant, null-terminated character string that MUST specify the name of the account to delete.

Note The above **LPCWSTR** field actually stores the offset to the actual string data. The string data MUST be located at the end of the structure array.

2.2.19 FAX_ACTIVITY_LOGGING_CONFIGW

The **FAX_ACTIVITY_LOGGING_CONFIGW** data type MAY be passed as an out parameter (as a byte array) in a **FAX_GetActivityLoggingConfiguration** call. Because this array is passed as a byte array, it requires custom marshaling. For reference, the data type definition is shown below:

```
typedef struct FAX_ACTIVITY_LOGGING_CONFIGW {
  DWORD dwSizeOfStruct;
  BOOL bLogIncoming;
  BOOL bLogOutgoing;
  LPWSTR lptstrDBPath;
} FAX_ACTIVITY_LOGGING_CONFIGW,
 *PFAX_ACTIVITY_LOGGING_CONFIGW;
```

dwSizeOfStruct: A DWORD that MUST specify the size of this structure.

bLogIncoming: A Boolean flag that MUST indicate whether incoming fax activities are logged.

bLogOutgoing: A Boolean flag that MUST indicate whether outgoing fax activities are logged.

lptstrDBPath: A pointer to a string that MUST identify the directory on the server where the **activity logging** database files reside. The directory SHOULD be created if it does not exist. The full path to the activity logging files MUST be [lptstrDBPath]\ACTIVITY_LOG_INBOX_FILE for the incoming activity and [lptstrDBPath]\ACTIVITY_LOG_OUTBOX_FILE for the outgoing activity.

Value	Meaning
ACTIVITY_LOG_INBOX_FILENAME	For InboxLOG
ACTIVITY_LOG_OUTBOX_FILENAME	For OutboxLOG

2.2.20 FAX_ARCHIVE_CONFIGW

The **FAX_ARCHIVE_CONFIGW** data type MAY be passed as an out parameter (as a byte array) in a **FAX_GetArchiveConfiguration** call. Because this array is passed as a byte array, it requires custom marshaling.

Note The **FAX_GetArchiveConfiguration** method call has been deprecated in Windows Vista. This method call and the **FAX_ARCHIVE_CONFIGW** data type have been replaced by **FAX_GENERAL_CONFIG**.

For reference, the data type definition is shown below:

```
typedef struct FAX_ARCHIVE_CONFIGW {
  DWORD dwSizeOfStruct;
  BOOL bUseArchive;
  LPWSTR lpcstrFolder;
  BOOL bSizeQuotaWarning;
  DWORD dwSizeQuotaHighWatermark;
  DWORD dwSizeQuotaLowWatermark;
  DWORD dwAgeLimit;
  DWORDLONG dwlArchiveSize;
} FAX_ARCHIVE_CONFIGW,
```

```
*PFAX_ARCHIVE_CONFIGW;
```

dwSizeOfStruct: A DWORD that MUST specify the size of the structure.

bUseArchive: A Boolean value that MUST specify whether archiving is turned on for the specified folder name.

lpcstrFolder: A pointer to the string that MUST be used to specify the folder name. This field MUST contain the offset to the location where the actual string is stored. The receiver SHOULD retrieve the string contents from the mentioned offset.

bSizeQuotaWarning: A Boolean flag that MUST indicate whether the fax server SHOULD issue an event log warning if the archive quota exceeds the watermarks defined by the **dwSizeQuotaHighWatermark** and **dwSizeQuotaLowWatermark** members.

dwSizeQuotaHighWatermark: A DWORD that MUST specify the high watermark of the archive size limit. If the size of the archive exceeds this value and if the **bSizeQuotaWarning** member is set to TRUE, an event log warning SHOULD be issued.

dwSizeQuotaLowWatermark: A DWORD that MUST specify the low watermark of the archive size limit. If an event log warning was already issued, no more events SHOULD be issued until the size of the archive drops below this value.

dwAgeLimit: A DWORD that MUST specify the number of days the fax server will keep fax messages in the archive. If a fax message stays in the archive longer than the value specified, it MAY be automatically deleted. If this value is zero, the time limit MUST not be used.

dwlArchiveSize: A DWORD that MUST specify the size, in bytes, of the archive.

In the above, the above **LPWSTR** field MUST store the offset to the actual string data. The string data MUST be located at the end of the structure.

2.2.21 FAX_CONFIGURATIONW

The **FAX_CONFIGURATIONW** structure MAY be passed as an out parameter (as a byte array) or in a **FAX_GetConfiguration** call. This structure MAY also be passed as an in parameter (as a byte array) in a **FAX_SetConfiguration** call. Because this array is passed as a byte array, it requires custom marshaling.

```
typedef struct _FAX_CONFIGURATIONW {
  DWORD SizeOfStruct;
  DWORD Retries;
  DWORD RetryDelay;
  DWORD DirtyDays;
  BOOL Branding;
  BOOL UseDeviceTsid;
  BOOL ServerCp;
  BOOL PauseServerQueue;
  FAX_TIME StartCheapTime;
  FAX_TIME StopCheapTime;
  BOOL ArchiveOutgoingFaxes;
  LPCWSTR ArchiveDirectory;
  LPCWSTR Reserved;
} FAX_CONFIGURATIONW,
```

```
*PFAX_CONFIGURATIONW;
```

SizeOfStruct: A DWORD that MUST specify the size, in bytes, of the FAX_CONFIGURATIONW structure. The calling application MUST set this member to sizeof(FAX_CONFIGURATIONW) before it calls the **FAX_SetConfiguration** method.

Retries: A DWORD variable that MUST specify the number of times the fax server SHOULD attempt to retransmit an outgoing fax if the initial transmission fails.

RetryDelay: A DWORD variable that MUST specify the number of minutes that SHOULD elapse between retransmission attempts by the fax server.

DirtyDays: A DWORD variable that MUST specify the number of days the fax server SHOULD retain an unsent job in the fax jobqueue. A transmission might not be sent, for example, if an invalid fax number or date is specified, or if the sending device receives a busy signal multiple times.

Branding: A Boolean flag that MUST indicate whether the fax server SHOULD generate a brand (banner) at the top of outgoing fax transmissions. If this member is TRUE, the fax server SHOULD generate a brand that contains transmission-related information like the transmitting station identifier, date, time, and page count.

UseDeviceTsid: A Boolean flag that MUST indicate whether the fax server will use the device's transmitting station identifier instead of the value specified in the TSID member of the FAX_JOB_PARAM structure. If this member is TRUE, the server MUST use the device's transmitting station identifier.

ServerCp: A Boolean flag that MUST indicate whether fax client applications MAY include a user-designed cover page with the fax transmission. If this member is TRUE, the client MUST use a common cover page stored on the fax server. If this member is FALSE, the client MAY use a personal cover page file.

PauseServerQueue: A Boolean flag that MUST indicate whether the fax server has paused the fax jobqueue. If this member is TRUE, the queue SHOULD be paused.

StartCheapTime: Specifies a **FAX_TIME** structure that MUST indicate the hour and minute at which the discount period begins. The discount period applies only to outgoing transmissions.

StopCheapTime: Specifies a **FAX_TIME** structure that MUST indicate the hour and minute at which the discount period ends. The discount period applies only to outgoing transmissions.

ArchiveOutgoingFaxes: A Boolean flag that MUST indicate whether the fax server SHOULD archive outgoing fax transmissions. If this member is TRUE, the server MUST archive outgoing transmissions in the directory specified by the **ArchiveDirectory** member.

ArchiveDirectory: A pointer to a constant, null-terminated character string that MUST specify the fully qualified path of the directory in which outgoing fax transmissions SHOULD be archived. The path MAY be a UNC path or a path that begins with a drive letter. The fax server MAY ignore this member if the **ArchiveOutgoingFaxes** member is FALSE. This member MAY be NULL if the **ArchiveOutgoingFaxes** member is FALSE.

Reserved: Reserved.

Note This MUST be set to NULL by the client.

The above structure MUST be passed as a byte array buffer.<1> The structure MUST be present at the start of the buffer. The **LPCWSTR** fields in the structure MUST store the offsets to the actual string data, which MUST be located at the end of the structure. The **LPCWSTR** strings located at the end of the buffer MUST be in the same order of occurrence in the structure.

2.2.22 FAX_DEVICE_PROVIDER_INFO

An array of this data type **FAX_DEVICE_PROVIDER_INFO** MAY be passed as an out parameter (as a byte array) in a **FAX_EnumerateProviders** call. Because this array is passed as a byte array, it requires custom marshaling.

```
typedef struct FAX_DEVICE_PROVIDER {
  DWORD wSizeOfStruct;
  LPCWSTR lpctstrFriendlyName;
  LPCWSTR lpctstrImageName;
  LPCWSTR lpctstrProviderName;
  LPCWSTR lpctstrGUID;
  DWORD dwCapabilities;
  FAX_VERSION Version;
  FAX_ENUM_PROVIDER_STATUS Status;
  DWORD dwLastError;
} FAX_DEVICE_PROVIDER,
 *PFAX_DEVICE_PROVIDER;
```

wSizeOfStruct: A DWORD that MUST specify the size, in bytes, of the structure.

lpctstrFriendlyName: A pointer to a constant string, which MUST contain the Fax Service Provider's (FSP) user-friendly name, suitable for display.

lpctstrImageName: A pointer to a constant string that MUST specify the full path and file name for the FSP execution components. In Windows, this points to the FSP DLL.

lpctstrProviderName: A pointer to a constant string that MUST specify the name of the telephony service provider associated with the devices for the FSP.

lpctstrGUID: A pointer to a constant string that MUST specify the Globally Unique Identifier (GUID) for the FSP.

dwCapabilities: A DWORD variable that MUST indicate the bitwise combination of capabilities of the FSP. This value MUST be set to zero.

Version: A **FAX_VERSION** variable that MUST indicate the version of the fax service execution components.

Status: A FAX_ENUM_PROVIDER_STATUS call which MUST specify the status of the FSP. For more information, see **FAX_ENUM_PROVIDER_STATUS**.

dwLastError: A DWORD that MUST specify the error code that was encountered while the provider was loaded and initialized.

In the preceding, each of the **LPCWSTR** fields MUST store the offset to the actual string data. The string data MUST be located at the end of the structure.

2.2.23 FAX_GENERAL_CONFIG

The **FAX_GENERAL_CONFIG** data type MAY be passed as an out parameter (as a byte array) in **FAX_GetGeneralConfiguration** call. Because this array is passed as a byte array, it requires custom marshaling. The size of the byte array MUST be 88 because 8 bytes are used for alignment and padding. Four bytes MUST be used for alignment before dwlArchiveSize field, and 4 bytes for padding at the end of the structure.

For reference, the data type definition is shown below:

```
typedef struct FAX_GENERAL_CONFIG {
  DWORD dwSizeOfStruct;
  BOOL  bUseArchive;
  LPCWSTR lpcwstrArchiveLocation;
  BOOL bSizeQuotaWarning;
  DWORD dwSizeQuotaHighWaterMark;
  DWORD dwSizeQuotaLowWaterMark;
  DWORD dwArchiveAgeLimit;
  DWORDLONG dwlArchiveSize;
  DWORD dwQueueAgeLimit;
  DWORD dwRetries;
  DWORD dwRetryDelay;
  BOOL bUseDeviceTSID;
  FAX_TIME dtDiscountStart;
  FAX_TIME dtDiscountEnd;
  BOOL bBranding;
  BOOL bAllowPersonalCP;
  DWORD dwQueueState;
  BOOL bAutoCreateAccountOnConnect;
  BOOL bIncomingFaxesArePublic;
} FAX_GENERAL_CONFIG,
 *PFAX_GENERAL_CONFIG;
```

dwSizeOfStruct: A DWORD value that MUST indicate the size, in bytes, of the structure.

bUseArchive: A Boolean value that MUST indicate whether the fax server uses an archive to store fax messages after they are successfully sent or received. If this member is TRUE, the fax server MUST archive fax messages.

lpcwstrArchiveLocation: A pointer to a null-terminated character string that indicates the archives folder location on the fax server file system. This string MUST not end in a backslash (\) character.

bSizeQuotaWarning: A Boolean value that MUST indicate whether the fax server MAY issue an event log warning if the archive quota exceeds the watermarks defined by the **dwSizeQuotaHighWatermark** and **dwSizeQuotaLowWatermark** members. If this member is TRUE, the fax server SHOULD issue a warning.

dwSizeQuotaHighWaterMark: A DWORD value that MUST indicate the high watermark of the archive size limit. If the size of the archive exceeds this value, and if the **bSizeQuotaWarning** member is set to TRUE, an event log warning SHOULD be issued.

dwSizeQuotaLowWaterMark: A DWORD value that MUST indicate the low watermark of the archive size limit. If the size of the archive falls below this value, and if the **bSizeQuotaWarning** member is set to TRUE, an event log warning SHOULD be issued.

dwArchiveAgeLimit: A DWORD value that MUST indicate the number of days messages are retained in the archives.

dwlArchiveSize: A DWORDLONG value that MUST indicate the actual size of the archives.

dwQueueAgeLimit: A DWORD value that MUST indicate the number of days unsent fax jobs are retained in the fax queue.

dwRetries: A DWORD value that MUST indicate the number of times the fax server will attempt to retransmit an outgoing fax if the initial transmission fails.

dwRetryDelay: A DWORD value that MUST indicate the minimum number of minutes that will elapse between retransmission attempts by the fax server.

bUseDeviceTSID: A Boolean value that MUST indicate whether the fax server uses the device's transmitting station identifier instead of the value that is specified when a new job is submitted to the server. If this member is TRUE, the server SHOULD use the device's transmitting station identifier.

dtDiscountStart: A **FAX_TIME** structure that MUST indicate the hour and minute at which the discount period begins. The discount period applies only to outgoing transmissions.

dtDiscountEnd: A **FAX_TIME** structure that MUST indicate the hour and minute at which the discount period ends. The discount period applies only to outgoing transmissions.

bBranding: A Boolean value that MUST specify whether the fax server should generate a brand (banner) at the top of outgoing fax transmissions. If this member is TRUE, the fax server SHOULD generate a brand that contains transmission-related information, such as the transmitting station identifier, date, time, and page count.

bAllowPersonalCP: A Boolean value that MUST indicate whether fax client applications can include a user-designed cover page with the fax transmission. If this member is TRUE, the client MAY provide a personal cover page file. If this member is FALSE, the client MUST use a common cover page stored on the fax server.

dwQueueState: A DWORD value that MUST contain state information about the fax server queue. If this value is zero, both the incoming and outgoing queues MUST be unblocked; otherwise, this value MUST be a combination of one or more of the following flags.

Value	Meaning
0x00000000	Both the incoming and outgoing queues are unblocked.
FAX_INCOMING_BLOCKED 0x00000001	Fax service will not receive new incoming faxes.
FAX_OUTBOX_BLOCKED 0x00000002	Fax service will reject submissions of new outgoing faxes to its queue.
FAX_OUTBOX_PAUSED 0x00000004	Fax service will not remove and execute outgoing fax jobs from its queue.

bAutoCreateAccountOnConnect: A Boolean value that MUST indicate whether the fax service automatically creates the fax account in response to a **FaxConnectFaxServer** call, provided the caller has some permission on the fax server. Administrators MAY set this member to TRUE to avoid pre-creating accounts for all users of the fax services. When it is set to FALSE,

and if the calling user does not have a fax account on the fax server, the **Fax_ConnectFaxServer** MUST fail with ERROR_ACCESS_DENIED.

bIncomingFaxesArePublic: A Boolean value that MUST indicate whether all incoming faxes can be viewed by everyone. When this setting is TRUE, all incoming faxes MAY be viewed by all users. When it is FALSE, only users whose accounts have FAX_ACCESS_MANAGE_RECEIVE_FOLDER permission MUST be able to view the incoming faxes.

In the preceding, the **LPCWSTR** field MUST store the offset to the actual string data. The string data MUST be located at the end of the structure.

2.2.24 FAX_GLOBAL_ROUTING_INFOW

An array of the **FAX_GLOBAL_ROUTING_INFOW** data type MAY be passed as an out parameter (as a byte array) in a **FAX_EnumGlobalRoutingInfo** call. Because this array is passed as a byte array, it requires custom marshaling.

For reference, the data type definition is shown below:

```
typedef struct FAX_GLOBAL_ROUTING_INFOW {
  DWORD SizeOfStruct;
  DWORD Priority;
  LPCWSTR Guid;
  LPCWSTR FriendlyName;
  LPCWSTR FunctionName;
  LPCWSTR ExtensionImageName;
  LPCWSTR ExtensionFriendlyName;
} FAX_GLOBAL_ROUTING_INFOW,
 *PFAX_GLOBAL_ROUTING_INFOW;
```

SizeOfStruct: A DWORD that MUST specify the size, in bytes, of the FAX_GLOBAL_ROUTING_INFO structure. The calling application MUST set this member to sizeof(FAX_GLOBAL_ROUTING_INFOW) before it calls the **FAX_SetGlobalRoutingInfo (section 3.1.4.81)** method.

Priority: Specifies a DWORD variable that MUST indicate the priority of the fax routing method. The priority determines the relative order in which the fax service calls the fax routing methods when the service receives a fax document. Values for this member MUST be 1 through n, where 1 is the highest priority.

Guid: A pointer to a constant, null-terminated character string that MUST specify the GUID that uniquely identifies the fax routing method of interest.

FriendlyName: A pointer to a constant, null-terminated character string that MUST specify the user-friendly name to display for the fax routing method.

FunctionName: A pointer to a null-terminated character string that MUST specify the name of the function that executes the specified fax routing method.

ExtensionImageName: A pointer to a constant, null-terminated character string that MUST specify the name of the fax routing extensions that implements the fax routing method.

ExtensionFriendlyName: A pointer to a constant, null-terminated character string that MUST specify the user-friendly name to display for the fax routing extensions that implements the fax routing method.

In the preceding, each of the **LPCWSTR** fields MUST store the offset to the actual string data. The string data MUST be located at the end of the structure array.

2.2.25 FAX_JOB_ENTRY_EX_1

An array of the **FAX_JOB_ENTRY_EX_1** (section 2.2.25) data type MAY be passed as an out parameter (as a byte array) in the **FAX_EnumJobsEx2 (section 3.1.4.23)**() call. The data type MAY also be passed as an out parameter in the **FAX_GetJobEx2 (section 3.1.4.43)**(). Because this array is passed as a byte array, it requires custom marshaling. For reference, the data type definition is shown below:

```
typedef struct FAX_JOB_ENTRY_EX_1 {
  DWORD dwSizeOfStruct;
  DWORD dwValidityMask;
  DWORDLONG dwlMessageId;
  DWORDLONG dwlBroadcastId;
  LPCWSTR lpctstrRecipientNumber;
  LPCWSTR  lpctstrRecipientName;
  LPCWSTR lpctstrSenderUserName;
  LPCWSTR lpctstrBillingCode;
  SYSTEMTIME tmOriginalScheduleTime;
  SYSTEMTIME tmSubmissionTime;
  FAX_ENUM_PRIORITY_TYPE Priority;
  DWORD dwDeliveryReportType;
  LPCWSTR lpctstrDocumentName;
  LPCWSTR lpctstrSubject;
  FAX_JOB_STATUS pStatus;
  BOOL  bHasCoverPage;
  LPCWSTR lpcwstrReceiptAddress;
  DWORD dwScheduleAction;
} FAX_JOB_ENTRY_EX_1,
 *PFAX_JOB_ENTRY_EX_1;
```

dwSizeOfStruct: A DWORD value that MUST indicate the size of the structure in bytes.

dwValidityMask: A DWORD value that MUST define a bitwise combination of valid fields in **FAX_ENUM_JOB_FIELDS (section 2.2.63)**.

dwlMessageId: A DWORDLONG value that MUST specify the unique identifier of the job.

dwlBroadcastId: A DWORDLONG value that MUST specify a unique identifier for the fax **broadcast** job. All outbound jobs created by the same broadcast operation share the same unique identifier.

lpctstrRecipientNumber: A pointer to a constant, null-terminated character string that MUST specify the fax number of the fax transmission recipient.

lpctstrRecipientName: A pointer to a constant, null-terminated character string that MUST specify the name of the fax transmission recipient.

lpctstrSenderUserName: A pointer to a constant, null-terminated character string that MUST specify the domain and user name of the sender of an outgoing fax job. Used for outgoing faxes only.

lpctstrBillingCode: A pointer to a constant, null-terminated character string that MUST specify an application-specific or server-specific billing code that applies to the fax transmission. Billing codes are optional. Used for outgoing faxes only.

tmOriginalScheduleTime: If the fax was sent using JSA_SPECIFIC_TIME, this member SHOULD specify a **SYSTEMTIME** structure that contains the date and time originally used to send the fax. The time specified MUST be expressed in Coordinated Universal Time (UTC). Used for outgoing faxes only.

tmSubmissionTime: A **SYSTEMTIME** structure that MUST contain the date and time the fax message was submitted for sending. The time specified MUST be expressed in UTC. Used for outgoing faxes only.

Priority: A FAX_ENUM_PRIORITY_TYPE (section 2.2.51) value that MUST contain the priority of the fax transmission. Used for outgoing faxes only.

dwDeliveryReportType: A DWORD value that MUST indicate the type of receipt that will be delivered to the sender when the fax is successfully sent or when the fax transmission fails. It MAY also specify whether a receipt will be sent for each recipient or for all of the recipients together. This member MUST be one of the following values defined in **FAX_ENUM_DELIVERY_REPORT_TYPES (section 2.2.62)**.

lpctstrDocumentName: A pointer to a constant, null-terminated character string that MUST specify a document name to associate with the fax document. This is the user-friendly name that appears in the print spooler. Used for outgoing faxes only.

lpctstrSubject: A pointer to a constant, null-terminated character string that MUST specify the subject used in the fax cover page. Used for outgoing faxes only.

pStatus: Pointer to a **FAX_JOB_STATUS (section 2.2.27)** structure that MUST contain the job's dynamic status information.

bHasCoverPage: Boolean value that MUST indicate whether the fax has a cover page. If this value is TRUE, the fax SHOULD have a cover page.

lpcwstrReceiptAddress: A pointer to a constant, null-terminated character string that MUST specify an e-mail address to which the fax service sends the delivery receipt when the job is finished. If the **DeliveryReportType** member is equal to DRT_EMAIL, the string is the address to which the DR or NDR SHOULD be sent. If the **DeliveryReportType** member is not equal to DRT_EMAIL, this member MUST be NULL.

dwScheduleAction: A DWORD value that MUST specify when to send the fax. This member MUST have one of the following values:

Value	Meaning
JSA_NOW 0x00000000	Send the fax as soon as a device is available.
JSA_SPECIFIC_TIME 0x00000001	Send the fax at the time specified by the **tmOriginalScheduleTime** member.
JSA_DISCOUNT_PERIOD	Send the fax during the discount rate period.

Value	Meaning
0x00000002	

In the above, each of the above **LPCWSTR** fields MUST store the offset to the actual string data. The string data MUST be located at the end of the structure array.

2.2.26 FAX_JOB_ENTRY_EXW

An array of the **FAX_JOB_ENTRY_EXW** (section 2.2.26) data type MAY be passed as an out parameter (as a byte array) in the **FAX_EnumJobsEx2 (section 3.1.4.23)** and **FAX_EnumJobsEx (section 3.1.4.22)** calls. The data type MAY also be passed as an out parameter in the **FAX_GetJobEx2 (section 3.1.4.43)** and **FAX_GetJobEx (section 3.1.4.42)** calls. Because this array is passed as a byte array, it requires custom marshaling.

For reference, the data type definition is shown below:

```
typedef struct FAX_JOB_ENTRY_EXW {
  DWORD dwSizeOfStruct;
  DWORD dwValidityMask;
  DWORDLONG dwlMessageId;
  DWORDLONG dwlBroadcastId;
  LPCWSTR lpctstrRecipientNumber;
  LPCWSTR  lpctstrRecipientName;
  LPCWSTR lpctstrSenderUserName;
  LPCWSTR lpctstrBillingCode;
  SYSTEMTIME tmOriginalScheduleTime;
  SYSTEMTIME tmSubmissionTime;
  FAX_ENUM_PRIORITY_TYPE Priority;
  DWORD dwDeliveryReportType;
  LPCWSTR lpctstrDocumentName;
  LPCWSTR lpctstrSubject;
  FAX_JOB_STATUS pStatus;
} FAX_JOB_ENTRY_EXW,
 *FAX_JOB_ENTRY_EXW;
```

dwSizeOfStruct: A DWORD value that MUST indicate the size of the structure in bytes.

dwValidityMask: A DWORD value that MUST define a bitwise combination of valid fields in **FAX_ENUM_JOB_FIELDS (section 2.2.63)**.

dwlMessageId: A DWORDLONG value that MUST specify the unique identifier of the job.

dwlBroadcastId: A DWORDLONG value that MUST specify a unique identifier for the fax broadcast job. All outbound jobs created by the same broadcast operation MUST share the same unique identifier.

lpctstrRecipientNumber: A pointer to a constant, null-terminated character string that MUST specify the fax number of the fax transmission recipient.

lpctstrRecipientName: A pointer to a constant, null-terminated character string that MUST specify the name of the fax transmission recipient.

lpctstrSenderUserName: A pointer to a constant, null-terminated character string that MUST specify the domain and user name of the sender of an outgoing fax job. Used for outgoing faxes only.

lpctstrBillingCode: A pointer to a constant, null-terminated character string that MUST specify an application-specific or server-specific billing code that applies to the fax transmission. Billing codes are optional. Used for outgoing faxes only.

tmOriginalScheduleTime: If the fax was sent using JSA_SPECIFIC_TIME, this member SHOULD specify a **SYSTEMTIME** structure that contains the date and time originally used to send the fax. The time specified must be expressed in Coordinated Universal Time (UTC). Used for outgoing faxes only.

tmSubmissionTime: A **SYSTEMTIME** structure that contains the date and time the fax message was submitted for sending. The time specified MUST be expressed in UTC. Used for outgoing faxes only.

Priority: A **FAX_ENUM_PRIORITY_TYPE (section 2.2.51)** value that contains the priority of the fax transmission. Used for outgoing faxes only.

dwDeliveryReportType: A DWORD value that MUST indicate the type of receipt that will be delivered to the sender when the fax is successfully sent or when the fax transmission fails. It MAY also specify whether a receipt will be sent for each recipient or for all of the recipients together. This member MUST have one of the values defined in **FAX_ENUM_DELIVERY_REPORT_TYPES (section 2.2.62)**.

lpctstrDocumentName: A pointer to a constant, null-terminated character string that MUST specify a document name to associate with the fax document. This MAY be the user-friendly name that appears in the print spooler. Used for outgoing faxes only.

lpctstrSubject: A pointer to a constant, null-terminated character string that MUST specify the subject used in the fax cover page. Used for outgoing faxes only.

pStatus: Pointer to a **FAX_JOB_STATUS (section 2.2.27)** structure that MUST contain the job's dynamic status information.

2.2.27 FAX_JOB_STATUS

The **FAX_JOB_STATUS** (section 2.2.27) data type MAY be passed as a pointer reference inside **FAX_JOB_ENTRY_EX (section 2.2.26)** or **FAX_JOB_ENTRY_EX1 (section 2.2.25)**. Either of the above is passed as a byte array, and therefore, **FAX_JOB_STATUS** also requires custom marshaling.

For reference, the data type definition is shown below:

```
typedef struct FAX_JOB_STATUS {
  DWORD dwSizeOfStruct;
  DWORD dwValidityMask;
  DWORD dwJobID;
  DWORD dwJobType;
  DWORD dwQueueStatus;
  DWORD dwExtendedStatus;
  LPCWSTR  lpctstrExtendedStatus;
  DWORD dwSize;
  DWORD dwPageCount;
  DWORD dwCurrentPage;
  LPCWSTR lpctstrTsid;
```

```
  LPCWSTR lpctstrCsid;
  SYSTEMTIME  tmScheduleTime;
  SYSTEMTIME tmTransmissionStartTime;
  SYSTEMTIME tmTransmissionEndTime;
  DWORD dwDeviceID;
  LPCWSTR lpctstrDeviceName;
  DWORD dwRetries;
  LPCWSTR lpctstrCallerID;
  LPCWSTR lpctstrRoutingInfo;
  DWORD dwAvailableJobOperations;
} FAX_JOB_STATUS,
 *PFAX_JOB_STATUS;
```

dwSizeOfStruct: A DWORD that MUST specify the size, in bytes, of the structure.

dwValidityMask: A DWORD value that MUST define a bitwise combination of valid fields in **FAX_ENUM_JOB_FIELDS (section 2.2.63)**.

dwJobID: A DWORD that MUST contain the session job identifier of the fax job.

dwJobType: A DWORD that MUST specify the type of the fax job. It MUST be one of the following:

Value	Meaning
JT_UNKNOWN 0x0001	The job type is unknown. This value indicates that the fax server has not yet scheduled the job.
JT_SEND 0x0002	The job is an outgoing fax transmission.
JT_RECEIVE 0x0004	The job is an incoming fax transmission.
JT_ROUTING 0x0008	The fax server tried to route the fax transmission, but routing failed. The fax server MAY attempt to route the job again.
JT_FAIL_RECEIVE 0x0010	The fax server failed to receive the job. This value is included for backward compatibility.
JT_BROADCAST 0x0020	The job is an outgoing broadcast message.

dwQueueStatus: A DWORD that MUST specify a set of bit flags that indicate the queue status of the fax job. It MUST be one of the following:

Value	Meaning
JS_PENDING 0x00000001	The fax job is in the queue and pending service.
JS_INPROGRESS 0x00000002	The fax job is in progress.
JS_DELETING	The fax server is deleting the fax job.

Value	Meaning
0x00000004	
JS_FAILED 0x00000008	The fax job failed.
JS_PAUSED 0x00000010	The fax server paused the fax job.
JS_NOLINE 0x00000020	There is no line available to send the fax. The fax server MAY send the transmission when a line is available.
JS_RETRYING 0x00000040	The fax job failed. The fax server MAY attempt to retransmit the fax after a specified interval.
JS_RETRIES_EXCEEDED 0x00000080	The fax server exceeded the maximum number of retransmission attempts allowed. The fax will not be sent.
JS_COMPLETED 0x00000100	The fax job is completed.
JS_CANCELED 0x00000200	The fax job is canceled.
JS_CANCELING 0x00000400	The fax job is canceling.
JS_ROUTING 0x00000800	The fax job is being routed.

dwExtendedStatus: A DWORD variable that MUST contain a fax-extended status code.

Value	Meaning
JS_EX_DISCONNECTED 0x00000001	The sender or the caller disconnected the fax call.
JS_EX_INITIALIZING 0x00000002	The device is initializing a call.
JS_EX_DIALING 0x00000003	The device is dialing a fax number.
JS_EX_TRANSMITTING 0x00000004	The device is sending a fax document.
JS_EX_ANSWERED 0x00000005	The device answered a new call.
JS_EX_RECEIVING 0x00000006	The device is receiving a fax document.
JS_EX_LINE_UNAVAILABLE 0x00000007	The device is not available because it is in use by another application.
JS_EX_BUSY 0x00000008	The device encountered a busy signal.

Value	Meaning
JS_EX_NO_ANSWER 0x00000009	The receiving device did not answer the call.
JS_EX_BAD_ADDRESS 0x0000000A	The device dialed an invalid fax number.
JS_EX_NO_DIAL_TONE 0x0000000B	The sending device cannot complete the call because it does not detect a dial tone.
JS_EX_FATAL_ERROR 0x0000000C	The device has encountered a fatal protocol error.
JS_EX_CALL_DELAYED 0x0000000D	The device delayed a fax call because the sending device received a busy signal multiple times. The device cannot retry the call because dialing restrictions exist (some countries/regions restrict the number of retry attempts when a number is busy).
JS_EX_CALL_BLACKLISTED 0x0000000E	The device could not complete a call because the telephone number was blocked or reserved; emergency numbers such as 911 are blocked.
JS_EX_NOT_FAX_CALL 0x0000000F	The device received a call that was a data call or a voice call.
JS_EX_PARTIALLY_RECEIVED 0x00000010	The incoming fax was partially received. Some (but not all) of the pages are available.
JS_EX_HANDLED 0x00000011	The fax service processed the outbound fax document; the fax service provider MAY transmit the document.
JS_EX_CALL_COMPLETED 0x00000012	The call was completed successfully.
JS_EX_CALL_ABORTED 0x00000013	The call was terminated.

lpctstrExtendedStatus: A pointer to a string that MUST contain a fax-extended status string provided by the Fax Service Provider (FSP). If this is NULL, **dwExtendedStatus** MUST be part of the predefined extended statuses. If it is not NULL, **dwExtendedStatus** MUST be the extended status code as provided by the FSP.

dwSize: A DWORD that MUST specify the size, in bytes, of the fax document to transmit.

dwPageCount: A DWORD that MUST indicate the total number of pages in the fax transmission.

dwCurrentPage: A DWORD that MUST indicate the page number in the fax transmission that the fax device is currently sending or receiving. The page count MUST be relative to one.

lpctstrTsid: A pointer to a constant identifier. This identifier MAY be a telephone number.

lpctstrCsid: A pointer to a constant, null-terminated character string that MUST specify the called station identifier. This identifier MAY be a telephone number.

tmScheduleTime: For outgoing faxes only. This member specifies a **SYSTEMTIME** structure that MUST specify the date and time to send the fax. The time specified MUST be expressed in Coordinated Universal Time (UTC).

tmTransmissionStartTime: A **SYSTEMTIME** structure that MUST specify the last date and time the fax message's transmission started. The time specified MUST be expressed in UTC.

tmTransmissionEndTime: A **SYSTEMTIME** structure that MUST specify the last date and time the fax message's transmission ended. The time specified MUST be expressed in UTC.

dwDeviceID: A DWORD that MUST specify the identifier of the device used to receive or send the fax document.

lpctstrDeviceName: A pointer to a string that MUST specify the name of the device used to receive or send the fax document.

dwRetries: For outgoing faxes only. This member is a DWORD that MUST specify the number of failed transmission retries counted for that fax job.

lpctstrCallerID: For incoming faxes only. This member is a pointer to a null-terminated character string that MUST identify the calling device that sent the active fax document. This string MAY include the telephone number of the calling device.

lpctstrRoutingInfo: For incoming faxes only. This member is a pointer to a null-terminated character string that SHOULD identify the routing information received by the active fax document. This string MAY include the telephone number of the called device.

dwAvailableJobOperations: A DWORD that MUST be a bitwise combination of values defined in **FAX_ENUM_JOB_OP (section 2.2.45)**. This value SHOULD be the possible operations available on the current job. Security considerations (that is, access rights of the caller) are not taken into account.

In the above, each of the above **LPCWSTR** fields MUST store the offset to the actual string data. The string data MUST be located at the end of the structure array.

2.2.28 FAX_MESSAGE_1

An array of the **FAX_MESSAGE_1** (section 2.2.28) data type MAY be passed as an out parameter (as a byte array) in the **FAX_EnumMessagesEx (section 3.1.4.25)** call. This data type MAY also be passed as an out parameter (as a byte array) in **FAX_GetMessageEx (section 3.1.4.46)**. Because this array is passed as a byte array, it requires custom marshaling.

For reference, the data type definition is shown below:

```
typedef struct FAX_MESSAGE_1 {
  DWORD dwSizeOfStruct;
  DWORD dwValidityMask;
  DWORDLONG dwlMessageId;
  DWORDLONG dwlBroadcastId;
  DWORD dwJobType;
  DWORD dwQueueStatus;
  DWORD dwExtendedStatus;
  LPCWSTR lpctstrExtendedStatus;
  DWORD dwSize;
  DWORD dwPageCount;
  LPCWSTR lpctstrRecipientNumber;
  LPCWSTR lpctstrRecipientName;
  LPCWSTR lpctstrSenderNumber;
  LPCWSTR lpctstrSenderName;
  LPCWSTR lpctstrTsid;
  LPCWSTR lpctstrCsid;
```

```
  LPCWSTR lpctstrSenderUserName;
  LPCWSTR lpctstrBillingCode;
  SYSTEMTIME tmOriginalScheduleTime;
  SYSTEMTIME tmSubmissionTime;
  SYSTEMTIME tmTransmissionStartTime;
  SYSTEMTIME tmTransmissionEndTime;
  LPCWSTR lpctstrDeviceName;
  FAX_ENUM_PRIORITY_TYPE Priority;
  DWORD dwRetries;
  LPCWSTR lpctstrDocumentName;
  LPCWSTR lpctstrSubject;
  LPCWSTR lpctstrCallerID;
  LPCWSTR lpctstrRoutingInfo;
  BOOL bHasCoverPage;
  DWORD dwReceiptType;
  LPCWSTR lpcwstrReceiptAddress;
  BOOL bServerReceiveFolder;
  DWORD dwMsgFlags;
} FAX_MESSAGE_1,
 *PFAX_MESSAGE_1;
```

dwSizeOfStruct: A DWORD that MUST specify the size of the structure in bytes.

dwValidityMask: A DWORD value that MUST define a bitwise combination of valid fields in **FAX_ENUM_JOB_FIELDS (section 2.2.63)**.

dwlMessageId: A DWORDLONG value that MUST specify the unique identifier of the fax message.

dwlBroadcastId: A DWORDLONG value that MUST specify a unique identifier for the fax broadcast job. All outbound jobs created by the same broadcast operation share the same unique identifier.

dwJobType: A DWORD that MUST specify the job type of the archived fax.

Value	Meaning
JT_SEND 0x0002	Job is an outgoing fax transmission.
JT_RECEIVE 0x0004	Job is an incoming fax transmission.

dwQueueStatus: A DWORD VALUE that MUST contain a set of bit flags that indicate the last queue status of the fax job, just before the message was archived.

Value	Meaning
JS_PENDING 0x00000001	Fax job is in the queue and awaiting service.
JS_INPROGRESS 0x00000002	Fax job is in progress.
JS_DELETING	Fax server is deleting the fax job.

Value	Meaning
0x00000004	
JS_FAILED 0x00000008	Fax job failed.
JS_PAUSED 0x00000010	The fax server paused the fax job.
JS_NOLINE 0x00000020	No line is available over which to send the fax. The fax server will send the transmission when a line is available.
JS_RETRYING 0x00000040	Fax job failed. The fax server will attempt to retransmit the fax after a specified interval.
JS_RETRIES_EXCEEDED 0x00000080	Fax server exceeded the maximum number of retransmission attempts allowed. The fax will not be sent.
JS_COMPLETED 0x00000100	Fax job is completed.
JS_CANCELED 0x00000200	Fax job is canceled.
JS_CANCELING 0x00000400	Fax job is being canceled.
JS_ROUTING 0x00000800	Fax job is being routed.

dwExtendedStatus: A DWORD value that MUST specify extended status information

Value	Meaning
JS_EX_DISCONNECTED 0x00000001	The sender or the caller disconnected the fax call.
JS_EX_INITIALIZING 0x00000002	The device is initializing a call.
JS_EX_DIALING 0x00000003	The device is dialing a fax number.
JS_EX_TRANSMITTING 0x00000004	The device is sending a fax document.
JS_EX_ANSWERED 0x00000005	The device answered a new call.
JS_EX_RECEIVING 0x00000006	The device is receiving a fax document.
JS_EX_LINE_UNAVAILABLE 0x00000007	The device is not available because it is in use by another application.
JS_EX_BUSY 0x00000008	The device encountered a busy signal.

Value	Meaning
JS_EX_NO_ANSWER 0x00000009	The receiving device did not answer the call.
JS_EX_BAD_ADDRESS 0x0000000A	The device dialed an invalid fax number.
JS_EX_NO_DIAL_TONE 0x0000000B	The sending device cannot complete the call because it does not detect a dial tone.
JS_EX_FATAL_ERROR 0x0000000C	The device has encountered a fatal protocol error.
JS_EX_CALL_DELAYED 0x0000000D	The device delayed a fax call because the sending device received a busy signal multiple times. The device cannot retry the call because dialing restrictions exist (some countries/regions restrict the number of retry attempts when a number is busy).
JS_EX_CALL_BLACKLISTED 0x0000000E	The device could not complete a call because the telephone number was blocked or reserved; emergency numbers such as 911 are blocked.
JS_EX_NOT_FAX_CALL 0x0000000F	The device received a call that was a data call or a voice call.
JS_EX_PARTIALLY_RECEIVED 0x00000010	The incoming fax was partially received. Some (but not all) of the pages are available.
JS_EX_HANDLED 0x00000011	The fax service processed the outbound fax document; the fax service provider will transmit the document.
JS_EX_CALL_COMPLETED 0x00000012	The call was completed successfully.
JS_EX_CALL_ABORTED 0x00000013	The call was terminated.

lpctstrExtendedStatus: A pointer to a constant, null-terminated character string that MUST specify a fax-extended status string provided by the fax service provider. If this member is NULL, **dwExtendedStatus** MUST be one of the predefined extended statuses. If the member is not NULL, **dwExtendedStatus** is the extended status code as provided by the fax service provider.

dwSize: A DWORD value that MUST specify the size, in bytes, of the fax document.

dwPageCount: A DWORD value that MUST indicate the total number of pages in the fax transmission.

lpctstrRecipientNumber: A pointer to a constant, null-terminated character string that MUST specify the fax number of the fax transmission recipient.

lpctstrRecipientName: A pointer to a constant, null-terminated character string that MUST specify the name of the fax transmission recipient.

lpctstrSenderNumber: A pointer to a constant, null-terminated character string that MUST specify the fax number of the fax transmission sender.

lpctstrSenderName: A pointer to a constant, null-terminated character string that MUST specify the name of the fax transmission sender.

lpctstrTsid: A pointer to a constant, null-terminated character string that MUST specify the transmitting station identifier. This identifier MAY be a telephone number.

lpctstrCsid: A pointer to a constant, null-terminated character string that MUST specify the called station identifier. This identifier is MAY be a telephone number.

lpctstrSenderUserName: A pointer to a constant, null-terminated character string that MUST specify the domain and user name of the sender of an outgoing fax job. Used for outgoing faxes only.

lpctstrBillingCode: A pointer to a constant, null-terminated character string that MUST specify an application-specific or server-specific billing code that applies to the fax transmission. Billing codes are optional. Used for outgoing faxes only.

tmOriginalScheduleTime: If the fax was sent using 1 (JSA_SPECIFIC_TIME), this member specifies a **SYSTEMTIME** structure that SHOULD specify the date and time originally used to send the fax. The time specified must be expressed in Coordinated Universal Time (UTC). Used for outgoing faxes only.

tmSubmissionTime: A **SYSTEMTIME** structure that MUST specify the date and time the fax message was submitted for sending. The time specified MUST be expressed in UTC. Used for outgoing faxes only.

tmTransmissionStartTime: A **SYSTEMTIME** structure that MUST specify the start date and time the fax message was last transmitted. The time specified MUST be expressed in UTC.

tmTransmissionEndTime: A **SYSTEMTIME** structure that MUST specify the end date and time the fax message was last transmitted. The time specified MUST be expressed in UTC.

lpctstrDeviceName: A pointer to a constant, null-terminated character string value that MUST specify the name of the device used to receive or send the fax document. The device MAY no longer exist.

Priority: A **FAX_ENUM_PRIORITY_TYPE (section 2.2.51)** value that contains the priority of the fax transmission. Used for outgoing faxes only.

dwRetries: A DWORD value that MUST specify the number of failed transmission retries counted for a fax job. Used for outgoing faxes only.

lpctstrDocumentName: A pointer to a constant, null-terminated character string that MUST specify a document name to associate with the fax document. This is the user-friendly name that appears in the print spooler. Used for outgoing faxes only.

lpctstrSubject: A pointer to a constant, null-terminated character string that MUST specify the subject used on the fax cover page. Used for outgoing faxes only.

lpctstrCallerID: A pointer to a null-terminated character string that MUST specify the calling device that sent the active fax document. This string MAY include the telephone number of the calling device. Used for incoming faxes only.

lpctstrRoutingInfo: A pointer to a null-terminated character string that MUST contain the routing information received by the active fax document. This string MAY include the telephone number of the called device. Used for incoming faxes only.

bHasCoverPage: Boolean value that MUST indicate whether the fax has a cover page. If this value is TRUE, the fax MAY have a cover page.

dwReceiptType: A DWORD value that MUST indicate the type of receipt delivered to the sender when a fax is successfully sent and when a fax transmission fails. It MAY also specify whether a receipt will be sent for each recipient or for all recipients. This member MUST be one of the following values:

Value	Meaning
frtNONE 0x00000000	No receipt is sent.
frtMAIL 0x00000001	Receipt is sent to the e-mail address specified in **lpcwstrReceiptAddress**. This receipt type provides detailed information on delivery status for each recipient.
frtMSGBOX 0x00000004	Receipt is sent to the sender by means of a message box on the sender's monitor. This receipt type indicates only how many transmissions were completed successfully and how many failed.

lpcwstrReceiptAddress: A pointer to a constant, null-terminated character string that MUST specify an e-mail address to which the fax service sends the delivery receipt when the job is finished.

bServerReceiveFolder: Boolean value that MUST indicate whether the fax has been assigned or if it should be sent to the server receive folder. If this value is TRUE, the fax is sent to the server receive folder. If it is FALSE, the fax is sent to the appropriate account.

dwMsgFlags: Bitmask that MUST indicate the state of various message flags. See **FAX_MESSAGE_PROPS (section 2.2.11)**.

Value	Meaning
FAX_MSG_FLAG_READ 0x00000001	Indicates whether this message is marked as read. The message is marked as read if this bit is set. The default is "unread" for inbox messages and "read" for sent messages.

In the above, each of the above **LPCWSTR** fields MUST store the offset to the actual string data. The string data MUST be located at the end of the structure array.

2.2.29 FAX_MESSAGE

An array of the **FAX_MESSAGE** (section 2.2.29) data type is passed as an out parameter (as a byte array) in the **FAX_EnumMessages (section 3.1.4.24)** call. This data type is also passed as an out parameter (as a byte array) in **FAX_GetMessage (section 3.1.4.45)**. Because this array is passed as a byte array, it requires custom marshaling.

For reference, the data type definition is shown below:

```
typedef struct FAX_MESSAGEW {
  DWORD dwSizeOfStruct;
  DWORD dwValidityMask;
  DWORDLONG dwlMessageId;
  DWORDLONG dwlBroadcastId;
  DWORD dwJobType;
  DWORD dwQueueStatus;
```

```
  DWORD dwExtendedStatus;
  LPCWSTR lpctstrExtendedStatus;
  DWORD dwSize;
  DWORD dwPageCount;
  LPCWSTR lpctstrRecipientNumber;
  LPCWSTR lpctstrRecipientName;
  LPCWSTR lpctstrSenderNumber;
  LPCWSTR lpctstrSenderName;
  LPCWSTR lpctstrTsid;
} FAX_MESSAGEW,
 *PFAX_MESSAGEW;
```

dwSizeOfStruct: A DWORD that MUST specify the size of the structure in bytes.

dwValidityMask: A DWORD value that MUST define a bitwise combination of valid fields in **FAX_ENUM_JOB_FIELDS (section 2.2.63)**.

dwlMessageId: A DWORDLONG value that MUST specify the unique identifier of the fax message.

dwlBroadcastId: A DWORDLONG value that MUST specify a unique identifier for the fax broadcast job. All outbound jobs created by the same broadcast operation share the same unique identifier.

dwJobType: A DWORD that MUST specify the job type of the archived fax.

Value	Meaning
JT_SEND 0x0002	Job is an outgoing fax transmission.
JT_RECEIVE 0x0004	Job is an incoming fax transmission.

dwQueueStatus: A DWORD VALUE that MUST contain a set of bit flags that indicate the last queue status of the fax job, just before the message was archived.

Value	Meaning
JS_PENDING 0x00000001	Fax job is in the queue and awaiting service.
JS_INPROGRESS 0x00000002	Fax job is in progress.
JS_DELETING 0x00000004	Fax server is deleting the fax job.
JS_FAILED 0x00000008	Fax job failed.
JS_PAUSED 0x00000010	The fax server paused the fax job.
JS_NOLINE	No line is available over which to send the fax. The fax server will send

Value	Meaning
0x00000020	the transmission when a line is available.
JS_RETRYING 0x00000040	Fax job failed. The fax server will attempt to retransmit the fax after a specified interval.
JS_RETRIES_EXCEEDED 0x00000080	Fax server exceeded the maximum number of retransmission attempts allowed. The fax will not be sent.
JS_COMPLETED 0x00000100	Fax job is completed.
JS_CANCELED 0x00000200	Fax job is canceled.
JS_CANCELING 0x00000400	Fax job is being canceled.
JS_ROUTING 0x00000800	Fax job is being routed.

dwExtendedStatus: A DWORD value that MUST specify extended status information.

Value	Meaning
JS_EX_DISCONNECTED 0x00000001	The sender or the caller disconnected the fax call.
JS_EX_INITIALIZING 0x00000002	The device is initializing a call.
JS_EX_DIALING 0x00000003	The device is dialing a fax number.
JS_EX_TRANSMITTING 0x00000004	The device is sending a fax document.
JS_EX_ANSWERED 0x00000005	The device answered a new call.
JS_EX_RECEIVING 0x00000006	The device is receiving a fax document.
JS_EX_LINE_UNAVAILABLE 0x00000007	The device is not available because it is in use by another application.
JS_EX_BUSY 0x00000008	The device encountered a busy signal.
JS_EX_NO_ANSWER 0x00000009	The receiving device did not answer the call.
JS_EX_BAD_ADDRESS 0x0000000A	The device dialed an invalid fax number.
JS_EX_NO_DIAL_TONE 0x0000000B	The sending device cannot complete the call because it does not detect a dial tone.

Value	Meaning
JS_EX_FATAL_ERROR 0x0000000C	The device has encountered a fatal protocol error.
JS_EX_CALL_DELAYED 0x0000000D	The device delayed a fax call because the sending device received a busy signal multiple times. The device cannot retry the call because dialing restrictions exist (some countries/regions restrict the number of retry attempts when a number is busy).
JS_EX_CALL_BLACKLISTED 0x0000000E	The device could not complete a call because the telephone number was blocked or reserved; emergency numbers such as 911 are blocked.
JS_EX_NOT_FAX_CALL 0x0000000F	The device received a call that was a data call or a voice call.
JS_EX_PARTIALLY_RECEIVED 0x00000010	The incoming fax was partially received. Some (but not all) of the pages are available.
JS_EX_HANDLED 0x00000011	The fax service processed the outbound fax document; the fax service provider will transmit the document.
JS_EX_CALL_COMPLETED 0x00000012	The call was completed successfully.
JS_EX_CALL_ABORTED 0x00000013	The call was terminated.

lpctstrExtendedStatus: A pointer to a constant, null-terminated character string that MUST specify a fax extended status string provided by the fax service provider. If this member is NULL, **dwExtendedStatus** MUST be one of the predefined extended statuses. If the member is not NULL, **dwExtendedStatus** is the extended status code as provided by the fax service provider.

dwSize: A DWORD value that MUST specify the size, in bytes, of the fax document.

dwPageCount: A DWORD value that MUST indicate the total number of pages in the fax transmission.

lpctstrRecipientNumber: A pointer to a constant, null-terminated character string that MUST specify the fax number of the fax transmission recipient.

lpctstrRecipientName: A pointer to a constant, null-terminated character string that MUST specify the name of the fax transmission recipient.

lpctstrSenderNumber: A pointer to a constant, null-terminated character string that MUST specify the fax number of the fax transmission sender.

lpctstrSenderName: A pointer to a constant, null-terminated character string that MUST specify the name of the fax transmission sender.

lpctstrTsid: A pointer to a constant, null-terminated character string that MUST specify the transmitting station identifier. This identifier MAY be a telephone number.

In the above, each of the above **LPCWSTR** fields MUST store the offset to the actual string data. The string data MUST be located at the end of the structure array.

2.2.30 RPC_FAX_OUTBOUND_ROUTING_GROUPW

An array of the **RPC_FAX_OUTBOUND_ROUTING_GROUPW** (section 2.2.30) data type MAY be passed as an out parameter (as a byte array) in the **FAX_EnumOutboundRoutingGroups (section 3.1.4.26)** call. Because this array is passed as a byte array, it requires custom marshaling.

For reference, the data type definition is shown below:

```
typedef struct FAX_OUTBOUND_ROUTING_GROUPW {
  DWORD dwSizeOfStruct;
  LPCWSTR lpctstrGroupName;
  DWORD dwNumDevices;
  LPDWORD lpdwDevices;
  FAX_ENUM_GROUP_STATUS Status;
} RPC_FAX_OUTBOUND_ROUTING_GROUPW,
 *PRPC_FAX_OUTBOUND_ROUTING_GROUPW;
```

dwSizeOfStruct: A DWORD value that MUST specify the size, in bytes, of this structure.

lpctstrGroupName: Unique string that MUST identify the group name. A valid group name MUST be a non-empty string that does not exceed 128 (MAX_ROUTING_GROUP_NAME) characters. The group name is expected to be case-insensitive.

dwNumDevices: A DWORD value that MUST indicate the number of devices in the group. The value MUST be in the range between 0 and 1,000.

lpdwDevices: A pointer to an array which MUST contain **dwNumDevices** entries. Each entry is a DWORD value that MUST indicate the device identifier in the group. A device MUST only appear once in a group's device list. A single device MAY belong to one or more groups. Groups MUST NOT be set to store invalid devices. The order of the devices in the array determines the order the devices SHOULD be used to send faxes, when the group is selected.

Status: Current status of the group from the enumeration **FAX_ENUM_GROUP_STATUS (section 2.2.46)**.

In the above, the above **LPCWSTR** fields MUST store the offset to the actual string data. The string data MUST be located at the end of the structure array. There is a pointer to a DWORD array containing the device ids.

2.2.31 RPC_FAX_OUTBOUND_ROUTING_RULEW

An array of the **RPC_FAX_OUTBOUND_ROUTING_RULEW** data type MAY be passed as an out parameter (as a byte array) in the **FAX_EnumOutboundRules (section 3.1.4.27)** call. Because this array is passed as a byte array, it requires custom marshaling.

For reference, the data type definition is shown below:

```
typedef struct FAX_OUTBOUND_ROUTING_RULEW {
  DWORD dwSizeOfStruct;
  DWORD dwAreaCode;
  DWORD dwCountryCode;
  LPWSTR lpwstrCountryName;
  union {
```

```
    DWORD dwDeviceId;
    LPCWSTR lpcstrGroupName;
  } Destination;
  BOOL bUseGroup;
} RPC_FAX_OUTBOUND_ROUTING_RULEW,
 *PFAX_OUTBOUND_ROUTING_RULEW;
```

dwSizeOfStruct: A DWORD that MUST specify the size of this structure.

dwAreaCode: A DWORD that MUST specify the area code of the rule. If this value is zero, the special (all other areas) value MUST be used. The **dwAreaCode** and **dwCountryCode** members MUST be a unique key.

dwCountryCode: A DWORD that MUST specify the country/region code of the rule. If this value is zero, the special (all other countries/regions) value MUST be used. The **dwAreaCode** and **dwCountryCode** members MUST be a unique key.

lpwstrCountryName: A pointer to a null-terminated string that MUST specify the country/region name. If **dwCountryCode** is zero, this value MUST be NULL. This is a read-only value.

Destination: Must indicate the destination of the rule. It MAY be either a device identifier or the name of an outbound routing group of devices.

bUseGroup: A Boolean value that MUST specify whether the group should be used in the destination. If TRUE, the group MUST be used as the rule's destination. If FALSE, the device MUST be used as the rule's destination.

In the above, each of the above **LPCWSTR** fields MUST store the offset to the actual string data. The string data MUST be located at the end of the structure array.

2.2.32 FAX_PRINTER_INFO

An array of the **FAX_PRINTER_INFO** data type MAY be passed as an out parameter (as a byte array) in the **FAX_GetServicePrinters (section 3.1.4.63)** call. Because this array is passed as a byte array, it requires custom marshaling.

For reference, the data type definition is shown below:

```
typedef struct FAX_PRINTER_INFOW {
  LPWSTR lptstrPrinterName;
  LPWSTR lptstrServerName;
  LPWSTR lptstrDriverName;
} FAX_PRINTER_INFOW,
 *PFAX_PRINTER_INFOW;
```

lptstrPrinterName: A pointer to a string that MUST contain the printer name.

lptstrServerName: A pointer to a string that MUST contain the name of the server to which the printer is attached.

lptstrDriverName: A pointer to a string that MUST contain the name of the printer driver for this printer.

In the above, each of the above **LPWSTR** fields MUST store the offset to the actual string data. The string data MUST be located at the end of the structure array.

2.2.33 FAX_PERSONAL_PROFILEW

The **FAX_PERSONAL_PROFILEW** structure defines the collection of items that MAY be received from a call to **FAX_GetPersonalProfileInfo (section 3.1.4.50)**. It defines the profile information on a fax sender or fax recipient stored at the server. Since this structure contains variable length string fields, it needs custom marshaling. This structure is defined below, explaining each of the entries inside it.

```
typedef struct FAX_PERSONAL_PROFILEW {
  DWORD dwSizeOfStruct;
  LPWSTR lptstrName;
  LPWSTR lptstrFaxNumbe;
  LPWSTR lptstrCompany;
  LPWSTR lptstrStreetAddress;
  LPWSTR  lptstrCity;
  LPWSTR lptstrState;
  LPWSTR lptstrZip;
  LPWSTR lptstrCountry;
  LPWSTR  lptstrTitle;
  LPWSTR lptstrDepartment;
  LPWSTR lptstrOfficeLocation;
  LPWSTR lptstrHomePhone;
  LPWSTR  lptstrOfficePhone;
  LPWSTR lptstrEmail;
  LPWSTR lptstrBillingCode;
  LPWSTR lptstrTSID;
} FAX_PERSONAL_PROFILEW,
 *PFAX_PERSONAL_PROFILEW,
 *LPCFAX_PERSONAL_PROFILEW;
```

dwSizeOfStruct: A DWORD that MUST specify the size, in bytes, of this structure.

lptstrName: A pointer to a string that MUST contain the recipient or sender name.

lptstrFaxNumbe: A pointer to a string that MUST contain the fax phone number associated with this profile.

lptstrCompany: A pointer to a string that MUST contain the name of the company at which the person associated with this profile works.

lptstrStreetAddress: A pointer to a string that MUST contain the street address associated with this profile.

lptstrCity: A pointer to a string that MUST contain the name of the city associated with this profile.

lptstrState: A pointer to a string that MUST contain the name of the state associated with this profile.

lptstrZip: A pointer to a string that MUST contain the zip code associated with this profile.

lptstrCountry: A pointer to a string that MUST contain the name of the country/region associated with this profile.

lptstrTitle: A pointer to a string that MUST contain the title of the person associated with this profile.

lptstrDepartment: A pointer to a string that MUST contain the department in which the person associated with this profile works.

lptstrOfficeLocation: A pointer to a string that MUST contain the office location of the person associated with this profile.

lptstrHomePhone: A pointer to a string that MUST contain the home phone number of the person associated with this profile.

lptstrOfficePhone: A pointer to a string that MUST contain the office phone number of the person associated with this profile.

lptstrEmail: A pointer to a string that MUST contain the e-mail address of the person associated with this profile.

lptstrBillingCode: A pointer to a string that MUST contain the billing code associated with this profile.

lptstrTSID: A pointer to a string that MUST contain the transmitting station identifier (TSID) associated with this profile.

The various elements of the structure are explained above. Note that the string data in the structure is stored toward the end of the structure and an offset to this string is actually stored in the **LPWSTR** field. So, in the byte stream for the structure, **LPWSTR** will indeed be an offset rather than a pointer.

2.2.34 FAX_PORT_INFO_EXW

The **FAX_PORT_INFO_EXW** structure defines information about a single fax device, known as a port. This structure MAY be used in calls like **FAX_EnumPortsEx (section 3.1.4.29)**, **FAX_SetPortEx (section 3.1.4.89)**, and **FAX_GetPortEx (section 3.1.4.52)**, which interact with the devices. **FAX_EnumPortsEx** (section 3.1.4.29) MAY return an array of this structure, while **FAX_GetPortEx** (section 3.1.4.52) and **FAX_SetPortEx** (section 3.1.4.89) work on individual devices. Because this structure has variable-length string fields and it is used as a byte stream, this structure requires custom marshaling. The definition of this structure is shown below, explaining each of the fields.

```
typedef struct FAX_PORT_INFO_EXW {
  DWORD dwSizeOfStruct;
  DWORD dwDeviceID;
  LPCWSTR lpctstrDeviceName;
  LPWSTR lptstrDescription;
  LPWSTR lpctstrProviderName;
  LPWSTR lpctstrProviderGUID;
  BOOL bSend;
  FAX_ENUM_DEVICE_RECEIVE_MODE ReceiveMode;
  DWORD dwStatus;
  DWORD dwRings;
  LPWSTR lptstrCsid;
```

```
  LPWSTR lptstrTsid;
} FAX_PORT_INFO_EXW,
 *PFAX_PORT_INFO_EXW;
```

dwSizeOfStruct: A DWORD specifying the size of this structure.

dwDeviceID: A DWORD that MUST specify the permanent line identifier for the specified fax device.

lpctstrDeviceName: A pointer to a constant, null-terminated character string that MUST specify the name of the fax device.

lptstrDescription: A pointer to a null-terminated character string that MUST specify the description of the fax device. This value cannot exceed 258 (MAX_FAX_STRING_LEN) characters.

lpctstrProviderName: A pointer to a constant, null-terminated character string that MUST specify the name of the fax device provider.

lpctstrProviderGUID: A pointer to a constant, null-terminated character string that MUST specify the GUID of the fax device provider.

bSend: A Boolean value that MUST specify whether the fax device is enabled to send faxes.

ReceiveMode: An enumerated value that MUST indicate whether the fax device is enabled to receive faxes and whether the calls are manually or automatically answered. This MUST be a value from the **FAX_ENUM_DEVICE_RECEIVE_MODE (section 2.2.42)** enumeration.

dwStatus: A DWORD that MUST specify the current status of the device. It SHOULD contain any combination of values from the **FAX_ENUM_DEVICE_STATUS (section 2.2.50)** enumeration.

dwRings: A DWORD that MUST specify the number of times an incoming fax call should ring before the specified device answers the call.

lptstrCsid: A pointer to a constant, null-terminated character string that MUST specify the called station identifier for faxes sent using this device. This identifier is usually a telephone number.

lptstrTsid: A pointer to a constant, null-terminated character string that MUST specify the transmitting station identifier for faxes sent using this device. This identifier is usually a telephone number.

In the above, each of the above **LPWSTR** fields MUST store the offset to the actual string data. The string data MUST be located at the end of the structure array.

2.2.35 FAX_RECEIPTS_CONFIGW

The **FAX_RECEIPTS_CONFIGW** structure defines the format for the receipt settings of the fax server. It MAY be used by **FAX_SetReceiptsConfiguration (section 3.1.4.91)** or **FAX_GetReceiptsConfiguration (section 3.1.4.54)** to set or get the configuration of receipts. Because the structure has variable-length string elements and because this structure is passed across as a byte data, it requires custom marshaling.

```
typedef struct FAX_RECEIPTS_CONFIGW {
```

```
  DWORD dwSizeOfStruct;
  DWORD dwAllowedReceipts;
  FAX_ENUM_SMTP_AUTH_OPTIONS SMTPAuthOption;
  LPWSTR lptstrReserved;
  LPWSTR lptstrSMTPServer;
  DWORD dwSMTPPort;
  LPWSTR lptstrSMTPFrom;
  LPWSTR lptstrSMTPUserName;
  LPWSTR lptstrSMTPPassword;
  BOOL bIsToUseForMSRouteThroughEmailMethod;
} FAX_RECEIPTS_CONFIGW,
 *PFAX_RECEIPTS_CONFIGW;
```

dwSizeOfStruct: A DWORD that MUST specify the size, in bytes, of the structure.

dwAllowedReceipts: A DWORD that MUST specify the type of receipts that the server supports. This member MUST be one of the values defined in **FAX_ENUM_DELIVERY_REPORT_TYPES (section 2.2.62)**.

SMTPAuthOption: A type of **SMTP** authentication that the server will use for SMTP connections. The options MUST be one of the enumerations defined in **FAX_ENUM_SMTP_AUTH_OPTIONS (section 2.2.43)**.

lptstrReserved: A pointer to a reserved value.

Note MUST be set to NULL by the client.

lptstrSMTPServer: A pointer to a string that MUST specify the SMTP server name.

dwSMTPPort: A DWORD that MUST specify the port number of the SMTP server.

lptstrSMTPFrom: A pointer to a string that MUST specify the SMTP e-mail address of the sender of the fax receipt messages.

lptstrSMTPUserName: A pointer to a string that MUST specify the user name to use for Basic-authenticated SMTP connections.

lptstrSMTPPassword: A pointer to a string that MUST specify the password to use for Basic-authenticated SMTP connections. For anonymous access, no user name and password is required. For Basic and Integrated authentication, a cleartext password is sent over the wire. It is for the server to use the password that depends on the authentication mode.

bIsToUseForMSRouteThroughEmailMethod: If set to TRUE, the Microsoft routing extension MUST use the DRT_EMAIL receipts settings to route incoming faxes by e-mail.

The various elements of the structure are explained above. Note that the string data in the structure is stored toward the end of the structure and an offset to this string is actually stored in the **LPWSTR** field. Therefore, in the byte stream for the structure, **LPWSTR** is an offset instead of a pointer.

2.2.36 FAX_ROUTING_EXTENSION_INFO

The **FAX_ROUTING_EXTENSION_INFO** data type defines the format in which the routing extensions are enumerated and MAY be returned by a call to **FAX_EnumRoutingExtensions (section 3.1.4.30)**. The call MAY return a pointer to an array of routing extensions, where each

element of the array is of this type. Because the data is returned as a byte array, custom marshaling is required.

For reference, the data type definition is shown below:

```
typedef struct FAX_ROUTING_EXTENSION_INFOW {
  DWORD dwSizeOfStruct;
  LPCWSTR lpctstrFriendlyName;
  LPCWSTR lpctstrImageName;
  LPCWSTR lpctstrExtensionName;
  FAX_VERSION Version;
  FAX_ENUM_PROVIDER_STATUS Status;
  DWORD dwLastError;
} FAX_ROUTING_EXTENSION_INFOW,
 *PFAX_ROUTING_EXTENSION_INFOW;
```

dwSizeOfStruct: A DWORD that MUST specify the size, in bytes, of the size of the structure.

lpctstrFriendlyName: A pointer to a constant string that MUST specify the Fax Service Provider (FSP) user-friendly name, suitable for display. In the byte representation, this field MUST have a byte offset toward the end of the array where the actual data string is stored.

lpctstrImageName: A pointer to a constant string that MUST specify the full path and file name for the FSP binary. In the byte representation, this field MUST have a byte offset toward the end of the array where the actual data string is stored.

lpctstrExtensionName: A pointer to a constant string that MUST specify the name of the telephony service provider associated with the devices for the FSP. In the byte representation, this field MUST have a byte offset toward the end of the array where the actual data string is stored.

Version: A **FAX_VERSION (section 2.2.17)** structure that MUST contain version information for the fax routing execution component. In Windows, this is version information of the fax routing extension's binary.

Status: MUST specify the load status of the fax extension. For more information, see **FAX_ENUM_PROVIDER_STATUS (section 2.2.44)**.

dwLastError: A DWORD value that MUST specify the Win32 error code that was encountered while the extension was loaded and initialized.

In the above, each of the above **LPCWSTR** fields MUST store the offset to the actual string data. The string data MUST be located at the end of the structure array.

2.2.37 FAX_TAPI_LINECOUNTRY_ENTRY

The **FAX_TAPI_LINECOUNTRY_ENTRY** data type defines the arrangement of data inside the **FAX_TAPI_LINECOUNTRY_LIST (section 2.2.38)** structure, which MAY be passed as an out parameter to **FAX_GetCountryList (section 3.1.4.37)**. This structure holds information about a specific country/region in the array of **FAX_TAPI_LINECOUNTRY_LIST** (section 2.2.38). Because this structure contains variable-length strings and it is passed as a byte array, custom marshaling is required.

For reference, the data type definition is shown below:

```
typedef struct FAX_TAPI_LINECOUNTRY_ENTRYW {
  DWORD dwCountryID;
  DWORD dwCountryCode;
  LPCWSTR lpctstrCountryName;
  LPCWSTR lpctstrLongDistanceRule;
} FAX_TAPI_LINECOUNTRY_ENTRYW,
 *PFAX_TAPI_LINECOUNTRY_ENTRYW;
```

dwCountryID: A DWORD that MUST specify the country/region identifier.

dwCountryCode: A DWORD that MUST specify the country/region code.

lpctstrCountryName: A pointer to the unique string that MUST specify the country/region name. This field MUST store an offset toward the end of the structure array where the actual country/region name is stored.

lpctstrLongDistanceRule: A pointer to a null-terminated string that MUST contain the dialing rule for directly-dialed calls to other areas in the same country/region. This field MUST store an offset toward the end of the structure array where the rule is stored.

In the above, the above **LPCWSTR** field MUST store the offset to the actual string data. The string data MUST be located at the end of the structure.

2.2.38 FAX_TAPI_LINECOUNTRY_LIST

The **FAX_TAPI_LINECOUNTRY_LIST** data type defines the structure that **FAX_GetCountryList (section 3.1.4.37)** MAY use to return the list of countries/regions from TAPI. The structure has a pointer to a list of countries/regions, with each country/region's data defined by a **FAX_TAPI_LINECOUNTRY_ENTRY (section 2.2.37)** structure. Because this structure holds a list of countries/regions whose size is variable, it requires custom marshaling.

For reference, the data type definition is shown below:

```
typedef struct FAX_TAPI_LINECOUNTRY_LISTW {
  DWORD dwNumCountries;
  PFAX_TAPI_LINECOUNTRY_ENTRY LineCountryEntries;
} FAX_TAPI_LINECOUNTRY_LISTW,
 *PFAX_TAPI_LINECOUNTRY_LISTW;
```

dwNumCountries: A DWORD that MUST specify the number of countries/regions in the list.

LineCountryEntries: A pointer to an array that MUST contain the actual list of countries/regions. In the byte format, this field MUST store an offset toward the end of the structure, where the list of countries/regions is stored as an array of structures of type **FAX_TAPI_LINECOUNTRY_ENTRY** (section 2.2.37).

In the above structure definition, the pointer **FAX_TAPI_LINECOUNTRY_LIST** MUST be an offset toward the end of the structure, where the actual list of countries/regions is stored.

2.2.39 Fax-Specific Errors

The following Fax-Specific Errors MAY be returned by the server to the client and are of data type DWORD.

Return value/code	Description
0x1B59 FAX_ERR_SRV_OUTOFMEMORY	The fax server failed to allocate memory.
0x1B5A FAX_ERR_GROUP_NOT_FOUND	The fax server failed to locate an outbound routing group by name.
0x1B5B FAX_ERR_BAD_GROUP_CONFIGURATION	The fax server encountered an outbound routing group with bad configuration.
0x1B5C FAX_ERR_GROUP_IN_USE	The fax server cannot remove an outbound routing group because it is in use by one or more outbound routing rules.
0x1B5D FAX_ERR_RULE_NOT_FOUND	The fax server failed to locate an outbound routing rule by country/region code and area code.
0x1B5E FAX_ERR_NOT_NTFS	The fax server cannot set an archive folder to a non-NTFS partition.
0x1B5F FAX_ERR_DIRECTORY_IN_USE	The fax server cannot use the same folder for both the inbox and the sent-items archives.
0x1B60 FAX_ERR_FILE_ACCESS_DENIED	The fax server cannot access the specified file or folder.
0x1B61 FAX_ERR_MESSAGE_NOT_FOUND	The fax server cannot find the job or message by its ID.
0x1B62 FAX_ERR_DEVICE_NUM_LIMIT_EXCEEDED	The fax server cannot complete the operation because the number of active fax devices allowed for this version of Windows was exceeded.
0x1B63 FAX_ERR_NOT_SUPPORTED_ON_THIS_SKU	The fax server cannot complete the operation because it is not supported for this version of Windows.
0x1B64 FAX_ERR_VERSION_MISMATCH	The fax client/server versions mismatch.
0x1B65 FAX_ERR_RECIPIENTS_LIMIT	The limit on the number of recipients for a single fax broadcast was reached.

2.2.40 FAX_ENUM_MSG_FLAGS

The **FAX_ENUM_MSG_FLAGS** enumeration defines the possible flags that specify the read, unread status of a fax message.

```
typedef  enum
{
  FAX_MSG_FLAG_READ = 0x00000001
} FAX_ENUM_MSG_FLAGS;
```

FAX_MSG_FLAG_READ: Indicates whether this message is marked as read. The message MUST be marked as read if this bit is set. The default MAY be "unread" for an inbox message and "read" for a sent message.

2.2.41 FAX_ENUM_RULE_STATUS

The **FAX_ENUM_RULE_STATUS** enumeration defines the possible status values for an outbound routing rule

```
typedef  enum
{
  FAX_RULE_STATUS_VALID = 0x00000000,
  FAX_RULE_STATUS_EMPTY_GROUP = 0x00000001,
  FAX_RULE_STATUS_ALL_GROUP_DEV_NOT_VALID = 0x00000002,
  FAX_RULE_STATUS_SOME_GROUP_DEV_NOT_VALID = 0x00000003,
  FAX_RULE_STATUS_BAD_DEVICE = 0x00000004
} FAX_ENUM_RULE_STATUS;
```

FAX_RULE_STATUS_VALID: Indicates this outbound routing rule is valid.

FAX_RULE_STATUS_EMPTY_GROUP: Indicates the rule's destination group has no devices.

FAX_RULE_STATUS_ALL_GROUP_DEV_NOT_VALID: Indicates the rule's destination group has valid devices.

FAX_RULE_STATUS_SOME_GROUP_DEV_NOT_VALID: Indicates the rule's destination group has some invalid devices.

FAX_RULE_STATUS_BAD_DEVICE: Indicates the rule's destination device is not valid.

2.2.42 FAX_ENUM_DEVICE_RECEIVE_MODE

The **FAX_ENUM_DEVICE_RECEIVE_MODE** enumeration constants describe the receive mode for a fax device.

```
typedef  enum
{
  FAX_DEVICE_RECEIVE_MODE_OFF = 0x00000000,
  FAX_DEVICE_RECEIVE_MODE_AUTO = 0x00000001,
  FAX_DEVICE_RECEIVE_MODE_MANUAL = 0x00000002
} FAX_ENUM_DEVICE_RECEIVE_MODE;
```

FAX_DEVICE_RECEIVE_MODE_OFF: Do not answer incoming calls.

FAX_DEVICE_RECEIVE_MODE_AUTO: Automatically answer incoming calls after the specified number of rings.

FAX_DEVICE_RECEIVE_MODE_MANUAL: Manually answer incoming calls.

2.2.43 FAX_ENUM_SMTP_AUTH_OPTIONS

The **FAX_ENUM_SMTP_AUTH_OPTIONS** enumeration defines the type of authentication used for SMTP connections.

```
typedef  enum
{
  FAX_SMTP_AUTH_ANONYMOUS = 0x00000000,
  FAX_SMTP_AUTH_BASIC = 0x00000001,
  FAX_SMTP_AUTH_NTLM = 0x00000002
} FAX_ENUM_SMTP_AUTH_OPTIONS;
```

FAX_SMTP_AUTH_ANONYMOUS: The server will send fax transmission receipts using a non-authenticated SMTP server. The server's name and port are defined in the **FAX_RECEIPTS_CONFIGW (section 2.2.35)** structure.

FAX_SMTP_AUTH_BASIC: The server will send fax transmission receipts using a basic (plain text) authenticated SMTP server. The server's name, port, user name, and password are defined in the **FAX_RECEIPTS_CONFIGW** (section 2.2.35) structure.

FAX_SMTP_AUTH_NTLM: The server will send fax transmission receipts using an NTLM-authenticated SMTP server. The server's name, port, user name, and password are defined in the **FAX_RECEIPTS_CONFIGW** (section 2.2.35) structure.

2.2.44 FAX_ENUM_PROVIDER_STATUS

The **FAX_ENUM_PROVIDER_STATUS** enumeration defines load status types for Fax Service Providers (FSPs).

```
typedef  enum
{
  FAX_PROVIDER_STATUS_SUCCESS = 0x00000000,
  FAX_PROVIDER_STATUS_SERVER_ERROR = 0x00000001,
  FAX_PROVIDER_STATUS_BAD_GUID = 0x00000002,
  FAX_PROVIDER_STATUS_BAD_VERSION = 0x00000003,
  FAX_PROVIDER_STATUS_CANT_LOAD = 0x00000004,
  FAX_PROVIDER_STATUS_CANT_LINK = 0x00000005,
  FAX_PROVIDER_STATUS_CANT_INIT = 0x00000006
} FAX_ENUM_PROVIDER_STATUS;
```

FAX_PROVIDER_STATUS_SUCCESS: The provider was successfully loaded, linked, and initialized.

FAX_PROVIDER_STATUS_SERVER_ERROR: Error encountered while trying to load, link, and initialize the provider. This is a server-related error. For more information about the error code, see the **dwLastError** member of the **FAX_DEVICE_PROVIDER_INFO (section 2.2.22)** or **FAX_ROUTING_EXTENSION_INFO (section 2.2.36)** structures.

FAX_PROVIDER_STATUS_BAD_GUID: Error encountered while parsing the installation data of the device provider. The Globally Unique Identifier (GUID) of the device provider is invalid.

FAX_PROVIDER_STATUS_BAD_VERSION: Error encountered while parsing the installation data of the device provider. The API version of the device provider is invalid.

FAX_PROVIDER_STATUS_CANT_LOAD: Error encountered while loading the provider's binary. Place the corresponding error in the **dwLastError** member of the **FAX_DEVICE_PROVIDER_INFO** (section 2.2.22) or **FAX_ROUTING_EXTENSION_INFO** (section 2.2.36) structures.

FAX_PROVIDER_STATUS_CANT_LINK: Error encountered while linking to external routines. Place the corresponding error in the dwLastError member of the FAX_DEVICE_PROVIDER_INFO (section 2.2.22) or FAX_ROUTING_EXTENSION_INFO (section 2.2.36) structures. In Windows, this indicates that an error was encountered while dynamically linking to one of the provider's DLL mandatory export functions.

FAX_PROVIDER_STATUS_CANT_INIT: Error encountered while calling the initialization function of the provider. Place the corresponding error in the **dwLastError** member of the **FAX_DEVICE_PROVIDER_INFO** (section 2.2.22) or **FAX_ROUTING_EXTENSION_INFO** (section 2.2.36) structures.

2.2.45 FAX_ENUM_JOB_OP

The **FAX_ENUM_JOB_OP** enumeration specifies the possible operations available on the current job. Security considerations (that is, access rights of the caller) are not taken into account.

```
typedef  enum
{
  FAX_JOB_OP_VIEW = 0x00000001,
  FAX_JOB_OP_PAUSE = 0x00000002,
  FAX_JOB_OP_RESUME = 0x00000004,
  FAX_JOB_OP_RESTART = 0x00000008,
  FAX_JOB_OP_DELETE = 0x00000010,
  FAX_JOB_OP_RECIPIENT_INFO = 0x00000020,
  FAX_JOB_OP_SENDER_INFO = 0x00000040
} FAX_ENUM_JOB_OP;
```

FAX_JOB_OP_VIEW: Indicates that the job can be viewed.

FAX_JOB_OP_PAUSE: Indicates that the job can be paused.

FAX_JOB_OP_RESUME: Indicates that the job can be resumed.

FAX_JOB_OP_RESTART: Indicates that the job can be restarted.

FAX_JOB_OP_DELETE: Indicates that the job can be deleted.

FAX_JOB_OP_RECIPIENT_INFO: Indicates that the job has recipient info.

FAX_JOB_OP_SENDER_INFO: Indicates that the job has sender info.

2.2.46 FAX_ENUM_GROUP_STATUS

The **FAX_ENUM_GROUP_STATUS** enumeration defines status types for outbound routing groups.

```
typedef  enum
{
  FAX_GROUP_STATUS_ALL_DEV_VALID = 0x00000000,
  FAX_GROUP_STATUS_EMPTY  = 0x00000001,
  FAX_GROUP_STATUS_ALL_DEV_NOT_VALID  = 0x00000002,
  FAX_GROUP_STATUS_SOME_DEV_NOT_VALID = 0x00000003
} FAX_ENUM_GROUP_STATUS;
```

FAX_GROUP_STATUS_ALL_DEV_VALID: All the devices in the group are valid and available for sending outgoing faxes.

FAX_GROUP_STATUS_EMPTY : The group is empty (does not contain any devices), and does not have any routing rules added.

FAX_GROUP_STATUS_ALL_DEV_NOT_VALID : All the devices in the group are not available for sending outgoing faxes. Devices may be unavailable if they do not exist or are offline.

FAX_GROUP_STATUS_SOME_DEV_NOT_VALID: Some (but not all) of the devices in the group are not available for sending outgoing faxes. Devices may be unavailable if they do not exist or are offline.

2.2.47 FAX_JOB_EXTENDED_STATUS_ENUM

The **FAX_JOB_EXTENDED_STATUS_ENUM** enumeration defines the extended status values for a fax job. These are basic values provided for developers of a fax service provider (FSP). However, with the exception of fjesPARTIALLY_RECEIVED, these values or other proprietary values that may be developed for a specific FSP are not recognized or interpreted by the fax server.

```
typedef  enum
{
  fjesNONE = 0,
  fjesDISCONNECTED = 1,
  fjesINITIALIZING = 2,
  fjesDIALING = 3,
  fjesTRANSMITTING = 4,
  fjesANSWERED = 5,
  fjesRECEIVING = 6,
  fjesLINE_UNAVAILABLE = 7,
  fjesBUSY = 8,
  fjesNO_ANSWER = 9,
  fjesBAD_ADDRESS = 10,
  fjesNO_DIAL_TONE = 11,
  fjesFATAL_ERROR = 12,
  fjesCALL_DELAYED = 13,
  fjesCALL_BLACKLISTED = 14,
  fjesNOT_FAX_CALL = 15,
  fjesPARTIALLY_RECEIVED = 16,
  fjesHANDLED = 17,
  fjesCALL_COMPLETED = 18,
  fjesCALL_ABORTED = 19,
  fjesPROPRIETARY = 0x01000000
} FAX_JOB_EXTENDED_STATUS_ENUM;
```

fjesNONE: No extended status value.

fjesDISCONNECTED: The sender or the caller disconnected the fax call.

fjesINITIALIZING: The device is initializing a call.

fjesDIALING: The device is dialing a fax number.

fjesTRANSMITTING: The device is sending a fax.

fjesANSWERED: The device answered a new call.

fjesRECEIVING: The device is receiving a fax.

fjesLINE_UNAVAILABLE: The device is not available because it is in use by another application.

fjesBUSY: The device encountered a busy signal.

fjesNO_ANSWER: The receiving device did not answer the call.

fjesBAD_ADDRESS: The device dialed an invalid fax number.

fjesNO_DIAL_TONE: The sending device cannot complete the call because it does not detect a dial tone.

fjesFATAL_ERROR: The device has encountered a fatal protocol error.

fjesCALL_DELAYED: The device delayed a fax call because the sending device received a busy signal multiple times. The device cannot retry the call because dialing restrictions exist (some countries/regions restrict the number of retry attempts when a number is busy).

fjesCALL_BLACKLISTED: The device could not complete a call because the telephone number was blocked or reserved; emergency numbers such as 911 are blocked.

fjesNOT_FAX_CALL: The device received a call that was a data call or a voice call.

fjesPARTIALLY_RECEIVED: The incoming fax was partially received. Some (but not all) of the pages are available.

fjesHANDLED: The fax service processed the outbound fax; the fax service provider will transmit the fax.

fjesCALL_COMPLETED: The call was completed.

fjesCALL_ABORTED: The call was aborted.

fjesPROPRIETARY: Obsolete.

2.2.48 FAX_TIME

The **FAX_TIME** structure represents a time, using individual members for the current hour and minute. The time is expressed in Coordinated Universal Time (UTC).

```
typedef struct FAX_TIME {
  WORD Hour;
  WORD Minute;
} FAX_TIME,
 *PFAX_TIME;
```

Hour: Specifies a 16-bit unsigned integer that is the current hour. Valid values are 0 through 23.

Minute: Specifies a 16-bit unsigned integer that is the current minute. Valid values are 0 through 59.

2.2.49 FAX_ENUM_EVENT_TYPE

The **FAX_ENUM_EVENT_TYPE** enumeration defines types of events that the caller can specify to receive.

```
typedef  enum
{
  FAX_EVENT_TYPE_LEGACY = 0x00000000,
  FAX_EVENT_TYPE_IN_QUEUE = 0x00000001,
  FAX_EVENT_TYPE_OUT_QUEUE = 0x00000002,
  FAX_EVENT_TYPE_CONFIG = 0x00000004,
  FAX_EVENT_TYPE_ACTIVITY = 0x00000008,
  FAX_EVENT_TYPE_QUEUE_STATE = 0x00000010,
  FAX_EVENT_TYPE_IN_ARCHIVE = 0x00000020,
  FAX_EVENT_TYPE_OUT_ARCHIVE = 0x00000040,
  FAX_EVENT_TYPE_FXSSVC_ENDED = 0x00000080,
  FAX_EVENT_TYPE_DEVICE_STATUS = 0x00000100,
  FAX_EVENT_TYPE_NEW_CALL = 0x00000200,
  FAX_EVENT_TYPE_LOCAL_ONLY = 0x80000000,
  FAX_EVENT_TYPE_INCOMING_CALL = 0x0100
} FAX_ENUM_EVENT_TYPE;
```

FAX_EVENT_TYPE_LEGACY: Signifies legacy events.

FAX_EVENT_TYPE_IN_QUEUE: Requests notification about fax jobs in the incoming queue. Whenever the state of an incoming fax job changes, a notification of that type is issued.

FAX_EVENT_TYPE_OUT_QUEUE: Requests notification about fax jobs in the outgoing queue. Whenever the state of an outgoing fax job changes, a notification of that type is issued.

FAX_EVENT_TYPE_CONFIG: Requests notifications about fax server configuration changes. Whenever the configuration of the fax server changes, a notification of that type is issued.

FAX_EVENT_TYPE_ACTIVITY: Requests notifications about the fax server activity. Whenever the activity state of the fax server changes, a notification of that type is issued.

FAX_EVENT_TYPE_QUEUE_STATE: Requests notifications about the fax queue state. Whenever the state of the fax server queue changes, a notification of that type is issued.

FAX_EVENT_TYPE_IN_ARCHIVE: Requests notifications about the removal of fax messages from the incoming messages archive. Whenever a message is removed from the archive, the archive type and the message unique identifier are issued in a notification message.

FAX_EVENT_TYPE_OUT_ARCHIVE: Requests notifications about the removal of fax messages from the outgoing messages archive. Whenever a message is removed from the archive, the archive type and the message unique identifier are issued in a notification message.

FAX_EVENT_TYPE_FXSSVC_ENDED: Specifies the shutdown of the fax server.

FAX_EVENT_TYPE_DEVICE_STATUS: Specifies that the status of a device has changed.

FAX_EVENT_TYPE_NEW_CALL: Specifies that a new incoming call was detected by the fax service.

FAX_EVENT_TYPE_LOCAL_ONLY: Specifies that the fax client (acting as Remote Procedure Call (RPC) server) wants to accept only local (same computer) notifications.

FAX_EVENT_TYPE_INCOMING_CALL: Specifies that an incoming call was detected by the fax service.

<2>

2.2.50 FAX_ENUM_DEVICE_STATUS

The **FAX_ENUM_DEVICE_STATUS** enumeration defines the possible status values of a fax device.

```
typedef  enum
{
  FAX_DEVICE_STATUS_POWERED_OFF = 0x00000001,
  FAX_DEVICE_STATUS_SENDING = 0x00000002,
  FAX_DEVICE_STATUS_RECEIVING = 0x00000004,
  FAX_DEVICE_STATUS_RINGING = 0x00000008
} FAX_ENUM_DEVICE_STATUS;
```

FAX_DEVICE_STATUS_POWERED_OFF: The device is powered off. This status MUST not be combined with any other status.

FAX_DEVICE_STATUS_SENDING: The device is currently sending one or more fax jobs.

FAX_DEVICE_STATUS_RECEIVING: The device is currently receiving one or more fax jobs.

FAX_DEVICE_STATUS_RINGING: The device is currently ringing.

2.2.51 FAX_ENUM_PRIORITY_TYPE

The **FAX_ENUM_PRIORITY_TYPE** enumeration defines types of priorities for outgoing faxes.

```
typedef  enum
{
  FAX_PRIORITY_TYPE_LOW = 0x00000000,
  FAX_PRIORITY_TYPE_NORMAL = 0x00000001,
  FAX_PRIORITY_TYPE_HIGH = 0x00000002
} FAX_ENUM_PRIORITY_TYPE;
```

FAX_PRIORITY_TYPE_LOW: The fax SHOULD be sent with a low priority.

FAX_PRIORITY_TYPE_NORMAL: The fax SHOULD be sent with a normal priority.

FAX_PRIORITY_TYPE_HIGH: The fax SHOULD be sent with a high priority.

2.2.52 FAX_EVENT

The **FAX_EVENT** structure represents the contents of an I/O completion packet. The fax server sends the completion packet to notify a fax client application about an asynchronous fax server event.

```
typedef struct _FAX_EVENT {
  DWORD SizeOfStruct;
  FILETIME TimeStamp;
  DWORD  DeviceId;
  DWORD EventId;
```

```
  DWORD JobId;
} FAX_EVENT,
 *PFAX_EVENT;
```

SizeOfStruct: Specifies the size, in bytes, of the **FAX_EVENT** structure. The fax server sets this member to **sizeof** (**FAX_EVENT**).

TimeStamp: Specifies a FILETIME structure, as specified in [MS-DTYP] section 2, that contains the time at which the fax server generated the event.

DeviceId: Specifies a **DWORD** variable that indicates the permanent line identifier for the fax device of interest.

EventId: Specifies a **DWORD** variable that identifies the current asynchronous event that occurred within the fax server. The following table lists the possible events and their meanings.

Value	Meaning
FEI_DIALING 0x00000001	The sending device is dialing a fax number.
FEI_SENDING 0x00000002	The sending device is transmitting a page of fax data.
FEI_RECEIVING 0x00000003	The receiving device is receiving a page of fax data.
FEI_COMPLETED 0x00000004	The device has completed a fax transmission call.
FEI_BUSY 0x00000005	The sending device has encountered a busy signal.
FEI_NO_ANSWER 0x00000006	The receiving device does not answer.
FEI_BAD_ADDRESS 0x00000007	The sending device cannot complete the call because the fax number is invalid.
FEI_NO_DIAL_TONE 0x00000008	The sending device cannot complete the call because it does not detect a dial tone.
FEI_DISCONNECTED 0x00000009	The device cannot complete the call because a fax device was disconnected or because the fax call itself was disconnected.
FEI_FATAL_ERROR 0x0000000A	The device encountered a fatal protocol error.
FEI_NOT_FAX_CALL 0x0000000B	The modem device received a data call or a voice call.
FEI_CALL_DELAYED 0x0000000C	The sending device received a busy signal multiple times. The device cannot retry the call because dialing restrictions exist (some countries/regions restrict the number of retry attempts when a number is busy).

Value	Meaning
FEI_CALL_BLACKLISTED 0x0000000D	The device cannot complete the call because the telephone number is blocked or reserved; numbers such as 911 are blocked.
FEI_RINGING 0x0000000E	The receiving device is ringing.
FEI_ABORTING 0x0000000F	The device is aborting a fax job.
FEI_ROUTING 0x00000010	The receiving device is routing a received fax document.
FEI_MODEM_POWERED_ON 0x000000011	The modem device was turned on.
FEI_MODEM_POWERED_OFF 0x000000012	The modem device was turned off.
FEI_IDLE 0x000000013	The device is idle.
FEI_FAXSVC_ENDED 0x000000014	The fax service has terminated. For more information, see the following Remarks section.
FEI_ANSWERED 0x000000015	The receiving device answered a new call.
FEI_JOB_QUEUED 0x000000016	The fax job has been queued.
FEI_DELETED 0x00000017	The fax job has been processed. The job identifier for the job is no longer valid.
FEI_INITIALIZING 0x00000018	The modem device is being initialized.
FEI_LINE_UNAVAILABLE 0x00000019	The device cannot complete the call because the requested line is unavailable.
FEI_HANDLED 0x0000001A	The fax job has been processed.
FEI_FAXSVC_STARTED 0x0000001B	The fax service has started. For more information, see the following Remarks section. Interchangeable with FEI_NEVENTS.
FEI_NEVENTS 0x0000001B	The total number of fax events received. For more information, see the following Remarks section. Interchangeable with FEI_FAXSVC_STARTED.

JobId: Specifies a unique number that identifies the fax job of interest. If this member is equal to the value 0xffffffff, it indicates an inactive fax job. Note that this number is not a print spooler identification number.

After a fax client application receives the FEI_FAXSVC_ENDED message from the fax service, it will no longer receive fax events. To resume receiving fax events, the application must call the **FaxInitializeEventQueue** function again when the fax service restarts. The application can determine whether the fax service is running by using the service control manager.

If the application receives events using notification messages, it can use the FEI_NEVENTS event. If the message is between the application's base window message and the base window message + FEI_NEVENTS, then the application can process the message as a fax window message. An application specifies the base window message using the *MessageStart* parameter to the **FaxInitializeEventQueue** function; the base window message must be greater than the WM_USER message.

2.2.53 FAX_EVENT_EX

The **FAX_EVENT_EX** structure defines information about asynchronous events delivered to applications that have registered to receive notification of fax events.

```
typedef struct {
  DWORD dwSizeOfStruct;
  FILETIME TimeStamp;
  FAX_ENUM_EVENT_TYPE EventType;
  union {
    FAX_EVENT_JOB JobInfo;
    FAX_ENUM_CONFIG_TYPE ConfigType;
    FAX_SERVER_ACTIVITY ActivityInfo;
    FAX_EVENT_NEW_CALL NewCall;
    DWORD dwQueueStates;
    FAX_EVENT_DEVICE_STATUS DeviceStatus;
  } EventInfo;
} FAX_EVENT_EX;
```

dwSizeOfStruct: A **DWORD** that contains the size, in bytes, of this structure.

TimeStamp: A FILETIME structure, as specified in [MS-DTYP] section 2, that contains the time the fax server generated the event.

EventType: One of the **FAX_ENUM_EVENT_TYPE (section 2.2.57)** values that indicates the type of event. Only a single bit is set in this value, thus there is notification for only a single event per value. This member defines which member of the **EventInfo** union is used. For the **FAX_EVENT_TYPE_FXSSVC_ENDED** (section 2.2.57) event, none of the **EventInfo** union members are used.

EventInfo: A union containing the information according to the event type.

JobInfo: For the **FAX_EVENT_TYPE_IN_QUEUE** (section 2.2.57), **FAX_EVENT_TYPE_OUT_QUEUE** (section 2.2.57), and **FAX_EVENT_TYPE_IN_ARCHIVE** (section 2.2.57) events, the **JobInfo** member contains a **FAX_EVENT_JOB (section 2.2.66)** structure with status about an existing job in the queue or archives.

ConfigType: For the **FAX_EVENT_TYPE_CONFIG (section 2.2.49)** event, the **ConfigType** member contains a **FAX_ENUM_CONFIG_TYPE (section 2.2.59)** enumeration value that indicates the type of the configuration that has changed. The receiver of this notification should call **FAX_GetConfiguration (section 3.1.4.36)** to get the new configuration.

ActivityInfo: For the **FAX_EVENT_TYPE_ACTIVITY** (section 2.2.49) event, the **ActivityInfo** member contains a **FAX_SERVER_ACTIVITY (section 2.2.15)** structure that contains information about the server activity that has changed. This event should

only be sent when the messages counters in the server activity structure change. No event is sent when a Windows event log entry is added on the server.

NewCall: For the **FAX_EVENT_TYPE_NEW_CALL** (section 2.2.49) event, the **NewCall** member contains a **FAX_EVENT_NEW_CALL (section 2.2.58)** structure that contains information about the new incoming call detected by the fax service. For more information, see **FAX_EVENT_NEW_CALL** (section 2.2.58).<3>

dwQueueStates: For the **FAX_EVENT_TYPE_QUEUE_STATE** (section 2.2.49) event, the **dwQueueStates** member contains the queue status. If this value is zero, both the incoming and outgoing queues are unblocked; otherwise, this value is a combination of one or more of the following values:

Value	Meaning
0x00000000	Both the incoming and outgoing queues are unblocked.
FAX_INCOMING_BLOCKED 0x00000001	The incoming faxes queue is blocked. The fax server will not answer any new incoming faxes.
FAX_OUTBOX_BLOCKED 0x00000002	The outbox queue is blocked. The fax server will not accept submission of new faxes. If the outbox is not paused, faxes in the queue are still being processed.
FAX_OUTBOX_PAUSED 0x00000004	The outbox queue is paused. The fax server will not start sending outgoing faxes from the queue. Fax transmissions in progress are not affected. If the outbox is not blocked, the fax server still accepts submission of new faxes to the queue.

DeviceStatus: For the **FAX_EVENT_TYPE_DEVICE_STATUS** (section 2.2.49) event, the **DeviceStatus** member contains a **FAX_EVENT_DEVICE_STATUS** (section 2.2.49)structure that indicates the status of the fax device.

2.2.54 FAX_EVENT_EX_1

The **FAX_EVENT_EX_1** structure defines information about asynchronous events delivered to applications that have been registered to receive notifications of fax events.

```
typedef struct {
  DWORD dwSizeOfStruct;
  FILETIME TimeStamp;
  FAX_ENUM_EVENT_TYPE EventType;
  union {
    FAX_EVENT_JOB_1 JobInfo;
    FAX_ENUM_CONFIG_TYPE ConfigType;
    FAX_SERVER_ACTIVITY ActivityInfo;
    FAX_EVENT_NEW_CALL NewCall;
    DWORD dwQueueStates;
    FAX_EVENT_DEVICE_STATUS DeviceStatus;
  } EventInfo;
} FAX_EVENT_EX_1,
 *PFAX_EVENT_EX_1;
```

dwSizeOfStruct: A **DWORD** containing the size, in bytes, of this structure.

TimeStamp: A **FILETIME** structure, as defined in [MS-DTYP] that contains the time when the fax server generated the event.

EventType: One of the **FAX_ENUM_EVENT_TYPE (section 2.2.49)** values that indicates the type of event. Only a single bit is set in this value, therefore, notification occurs for only a single event per value. This member defines which member of the **EventInfo** union is used. For the **FAX_EVENT_TYPE_FXSSVC_ENDED** (section 2.2.49) event, none of the **EventInfo** union members are used.

EventInfo: A union containing the information according to the event type.

JobInfo: For the **FAX_EVENT_TYPE_IN_QUEUE (section 2.2.56)**, **FAX_EVENT_TYPE_OUT_QUEUE** (section 2.2.56), and **FAX_EVENT_TYPE_IN_ARCHIVE** (section 2.2.56) events, the **JobInfo** member contains a **FAX_EVENT_JOB_1** (section 2.2.56) structure with status information about an existing job in the queue or archives.

ConfigType: For the **FAX_EVENT_TYPE_CONFIG** (section 2.2.49) event, the **ConfigType** member contains a **FAX_ENUM_CONFIG_TYPE (section 2.2.59)** enumeration value that indicates the type of the configuration that has changed. The receiver of this notification should call **FAX_GetConfiguration** to get the new configuration.

ActivityInfo: For the **FAX_EVENT_TYPE_ACTIVITY** (section 2.2.49) event, the **ActivityInfo** member contains a **FAX_SERVER_ACTIVITY (section 2.2.15)** structure that contains information about the server activity that has changed. This event should be sent only when the message counters in the server activity structure change. No event is sent when a Windows event log entry is added on the server.

NewCall: For the **FAX_EVENT_TYPE_NEW_CALL** (section 2.2.49) event, the **NewCall** member contains a **FAX_EVENT_NEW_CALL** (section 2.2.49) structure that contains information about the new incoming call detected by the fax service.

dwQueueStates: For the **FAX_EVENT_TYPE_QUEUE_STATE** (section 2.2.49) event, the **dwQueueStates** member contains the queue status. If this value is zero, both the incoming and outgoing queues are unblocked; otherwise, this value is a combination of one or more of the following values:

Value	Meaning
0x00000000	Both the incoming and outgoing queues are unblocked.
FAX_INCOMING_BLOCKED 0x00000001	The incoming faxes queue is blocked. The fax server will not answer any new incoming faxes.
FAX_OUTBOX_BLOCKED 0x00000002	The outbox queue is blocked. The fax server will not accept submission of new faxes. If the outbox is not paused, faxes in the queue are still being processed.
FAX_OUTBOX_PAUSED 0x00000004	The outbox queue is paused. The fax server will not start sending outgoing faxes from the queue. Fax transmissions in progress are not affected. If the outbox is not blocked, the fax server still accepts submission of new faxes to the queue.

DeviceStatus: For the **FAX_EVENT_TYPE_DEVICE_STATUS** (section 2.2.49) event, the **DeviceStatus** member contains a **FAX_EVENT_DEVICE_STATUS (section 2.2.55)** structure that indicates the status of the fax devices.

2.2.55 FAX_EVENT_DEVICE_STATUS

The **FAX_EVENT_DEVICE_STATUS** structure defines information about the status of a fax device.

```
typedef struct {
  DWORD dwDeviceId;
  DWORD dwNewStatus;
} FAX_EVENT_DEVICE_STATUS,
 *PFAX_EVENT_DEVICE_STATUS;
```

dwDeviceId: A **DWORD** value indicating the identification number of the device that had a status change.

dwNewStatus: A **DWORD** value indicating the new status. The value is a combination of values from **FAX_ENUM_DEVICE_STATUS (section 2.2.50)**.

2.2.56 FAX_EVENT_JOB_1

The **FAX_EVENT_JOB_1** structure defines information about notifications regarding a single job in the server's queue.

```
typedef struct {
  DWORDLONG dwlMessageId;
  FAX_ENUM_JOB_EVENT_TYPE Type;
  PFAX_JOB_STATUS pJobData;
  BOOL bServerReceiveFolder;
} FAX_EVENT_JOB_1,
 *PFAX_EVENT_JOB_1;
```

dwlMessageId: A **DWORDLONG** value that contains the unique identifier of the job.

Type: Specifies the type of information about the job. This will be one of the **FAX_ENUM_JOB_EVENT_TYPE (section 2.2.57)** enumeration values.

pJobData: If the **Type** member contains the FAX_JOB_EVENT_TYPE_STATUS value from the **FAX_ENUM_JOB_EVENT_TYPE** (section 2.2.57) enumeration, this member contains a pointer to a **FAX_JOB_STATUS (section 2.2.27)** structure that contains the current status of the job. Otherwise, this member is NULL.

bServerReceiveFolder: A **BOOL** value that indicates whether the job is still in the server's receive folder.

Value	Meaning
TRUE	The job is still in the server's receive folder; it has not been reassigned yet.
FALSE	The job has been reassigned and is no longer in the server's receive folder.

2.2.57 FAX_ENUM_JOB_EVENT_TYPE

The **FAX_ENUM_JOB_EVENT_TYPE** enumeration defines types of events for a single job.

```
typedef  enum
{
  FAX_JOB_EVENT_TYPE_ADDED = 0x00000000,
  FAX_JOB_EVENT_TYPE_REMOVED = 0x00000001,
  FAX_JOB_EVENT_TYPE_STATUS = 0x00000002,
  FAX_JOB_EVENT_TYPE_CHANGED = 0x00000003
} FAX_ENUM_JOB_EVENT_TYPE;
```

FAX_JOB_EVENT_TYPE_ADDED: A job was added to the queue or a message was added to the archive.

FAX_JOB_EVENT_TYPE_REMOVED: A job was removed from the queue or a message was removed from the archive.

FAX_JOB_EVENT_TYPE_STATUS: The job has changed its status. This does not apply to archive messages.

FAX_JOB_EVENT_TYPE_CHANGED: An archives message has changed.

2.2.58 FAX_EVENT_NEW_CALL

The **FAX_EVENT_NEW_CALL** structure defines notifications regarding a new incoming call.

```
typedef struct {
  HCALL hCall;
  DWORD dwDeviceId;
  LPTSTR lptstrCallerId;
} FAX_EVENT_NEW_CALLW,
 *PFAX_EVENT_NEW_CALLW;
```

hCall: Call **handle** of the new incoming call.

dwDeviceId: Identifier of the fax device on which the new incoming call has arrived.

lptstrCallerId: Pointer to a null-terminated Unicode string that contains the caller ID for the incoming call. It is set to **NULL** if no caller ID information is available.

2.2.59 FAX_ENUM_CONFIG_TYPE

The **FAX_ENUM_CONFIG_TYPE** enumeration indicates the type of configuration that has changed during a **FAX_ENUM_EVENT_TYPE (section 2.2.49)** event.

```
typedef  enum
{
  FAX_CONFIG_TYPE_RECEIPTS = 0x00000000,
  FAX_CONFIG_TYPE_ACTIVITY_LOGGING = 0x00000001,
  FAX_CONFIG_TYPE_OUTBOX = 0x00000002,
  FAX_CONFIG_TYPE_SENTITEMS = 0x00000003,
  FAX_CONFIG_TYPE_INBOX = 0x00000004,
  FAX_CONFIG_TYPE_SECURITY = 0x00000005,
  FAX_CONFIG_TYPE_EVENTLOGS = 0x00000006,
  FAX_CONFIG_TYPE_DEVICES = 0x00000007,
  FAX_CONFIG_TYPE_OUT_GROUPS = 0x00000008,
```

```
  FAX_CONFIG_TYPE_OUT_RULES = 0x00000009,
  FAX_CONFIG_TYPE_GENERAL_CONFIG = 0x0000000A
} FAX_ENUM_CONFIG_TYPE;
```

FAX_CONFIG_TYPE_RECEIPTS: The receipt configuration has changed.

FAX_CONFIG_TYPE_ACTIVITY_LOGGING: The activity logging configuration has changed.

FAX_CONFIG_TYPE_OUTBOX: The outbox configuration has changed.

FAX_CONFIG_TYPE_SENTITEMS: The sent items archive configuration has changed.

FAX_CONFIG_TYPE_INBOX: The Inbox configuration has changed.

FAX_CONFIG_TYPE_SECURITY: The security configuration has changed.

FAX_CONFIG_TYPE_EVENTLOGS: The event log configuration has changed.

FAX_CONFIG_TYPE_DEVICES: The device configuration has changed.

FAX_CONFIG_TYPE_OUT_GROUPS: The outbound routing groups configuration has changed.

FAX_CONFIG_TYPE_OUT_RULES: The outbound routing rules configuration has changed.

FAX_CONFIG_TYPE_GENERAL_CONFIG: The **general configuration** has changed.

2.2.60 FAX Data Types

The **FAX Data Types** for the fax server and client interfaces.

```
typedef [context_handle] HANDLE* RPC_FAX_HANDLE;

typedef [ref] RPC_FAX_HANDLE* PRPC_FAX_HANDLE;

typedef [context_handle] HANDLE RPC_FAX_PORT_HANDLE;

typedef [ref] RPC_FAX_PORT_HANDLE* PRPC_FAX_PORT_HANDLE;

typedef [context_handle] HANDLE* RPC_FAX_SVC_HANDLE;

typedef [ref] RPC_FAX_SVC_HANDLE* PRPC_FAX_SVC_HANDLE;

typedef [context_handle] HANDLE* RPC_FAX_MSG_ENUM_HANDLE;

typedef [ref] RPC_FAX_MSG_ENUM_HANDLE* PRPC_FAX_MSG_ENUM_HANDLE;

typedef [context_handle] HANDLE* RPC_FAX_COPY_HANDLE;

typedef [ref] RPC_FAX_COPY_HANDLE* PRPC_FAX_COPY_HANDLE;

typedef [context_handle] HANDLE* RPC_FAX_EVENT_HANDLE;

typedef [ref] RPC_FAX_EVENT_HANDLE* PRPC_FAX_EVENT_HANDLE;

typedef [context_handle] HANDLE* RPC_FAX_EVENT_EX_HANDLE;
```

```
typedef [ref] RPC_FAX_EVENT_EX_HANDLE* PRPC_FAX_EVENT_EX_HANDLE;
#ifdef SERVER STUB
typedef [range((0,16384))] DWORD RANGED_DWORD, *LPRANGED_DWORD;
#else
typedef DWORD* LPRANGED_DWORD;
#endif
```

RPC_FAX_HANDLE

A context handle used in fax client interfaces.

PRPC_FAX_HANDLE

A pointer to a context handle that is used in fax client interfaces.

RPC_FAX_PORT_HANDLE

An RPC context handle that references a specified fax port.

PRPC_FAX_PORT_HANDLE

A pointer to a context handle that references a specified fax port.

RPC_FAX_SVC_HANDLE

A fax service context handle.

PRPC_FAX_SVC_HANDLE

A pointer to a fax service context handle.

RPC_FAX_MSG_ENUM_HANDLE

A message enumeration handle.

PRPC_FAX_MSG_ENUM_HANDLE

A pointer to a message enumeration handle.

RPC_FAX_COPY_HANDLE

A context handle for a file.

PRPC_FAX_COPY_HANDLE

A pointer to a context handle for a file.

RPC_FAX_EVENT_HANDLE

A fax notification context handle.

PRPC_FAX_EVENT_HANDLE

A pointer to a fax notification context handle.

RPC_FAX_EVENT_EX_HANDLE

A fax notification context handle.

PRPC_FAX_EVENT_EX_HANDLE

A pointer to a fax notification context handle.

RANGED_DWORD

LPRANGED_DWORD

2.2.61 PRODUCT_SKU_TYPE

The **PRODUCT_SKU_TYPE** enumeration provides values that identify the different Stock-Keeping Unit (SKU) versions of an operating system.

```
typedef  enum
{
  PRODUCT_SKU_UNKNOWN = 0x00000000,
  PRODUCT_SKU_PERSONAL = 0x00000001,
  PRODUCT_SKU_PROFESSIONAL = 0x00000002,
  PRODUCT_SKU_SERVER = 0x00000004,
  PRODUCT_SKU_ADVANCED_SERVER = 0x00000008,
  PRODUCT_SKU_DATA_CENTER = 0x00000010,
  PRODUCT_SKU_DESKTOP_EMBEDDED = 0x00000020,
  PRODUCT_SKU_SERVER_EMBEDDED = 0x00000040,
  PRODUCT_SKU_WEB_SERVER = 0x00000080
} PRODUCT_SKU_TYPE;
```

PRODUCT_SKU_UNKNOWN: SKU of the operating system is unknown.

PRODUCT_SKU_PERSONAL: SKU of the operating system is Windows XP, Personal Edition.

PRODUCT_SKU_PROFESSIONAL: SKU of the operating system is Windows XP Professional Edition.

PRODUCT_SKU_SERVER: SKU of the operating system is Windows Server 2003, Standard Edition.

PRODUCT_SKU_ADVANCED_SERVER: SKU of the operating system is Windows Server 2003 Advanced Edition.

PRODUCT_SKU_DATA_CENTER: SKU of the operating system is Windows Server 2003, Datacenter Edition.

PRODUCT_SKU_DESKTOP_EMBEDDED: SKU of the operating system is Windows XP Embedded Edition.

PRODUCT_SKU_SERVER_EMBEDDED: SKU of the operating system is Windows Server 2003 Embedded Edition.

PRODUCT_SKU_WEB_SERVER: SKU of the operating system is Windows Server 2003 Web Server Edition.

2.2.62 FAX_ENUM_DELIVERY_REPORT_TYPES

The **FAX_ENUM_DELIVERY_REPORT_TYPE** enumeration defines the type of receipt delivered to the sender when the fax is successfully sent and when the fax transmission fails. It MAY also specify if a receipt will be sent for each recipient or for all the recipients together. The value of this parameter MUST be a logical combination of one of the delivery method flags and any of the delivery grouping flags.

```
typedef  enum
{
  DRT_NONE = 0x00000000,
  DRT_EMAIL = 0x00000001,
  DRT_UNUSED = 0x00000002,
  DRT_MSGBOX = 0x00000004,
  DRT_GRP_PARENT = 0x00000008,
  DRT_ATTACH_FAX = 0x00000010
} FAX_ENUM_DELIVERY_REPORT_TYPES;
```

DRT_NONE: Do not send a receipt.

DRT_EMAIL: Send the receipt by e-mail. The e-mail address will be the e-mail address of the sender.

DRT_UNUSED: This is unused.<4>

DRT_MSGBOX: Notification of the transmission result using a message box to the sending user's machine.<5>

DRT_GRP_PARENT: Delivery grouping flag. The format of the receipt is dependent on the delivery method. DRT_EMAIL will provide a detailed status for each recipient. DRT_MSGBOX will only indicate the number of recipients for which the transmission completed successfully and the number of recipients for which the transmission failed. If this flag is not sent, the receipt SHOULD be sent for each recipient.

DRT_ATTACH_FAX: Attach the fax Tagged Image File Format (TIFF) file to the receipt.

2.2.63 FAX_ENUM_JOB_FIELDS

The **FAX_ENUM_JOB_FIELDS** enumeration defines bit fields of valid fields in a job or message structure.

```
typedef  enum
{
  FAX_JOB_FIELD_JOB_ID = 0x00000001,
  FAX_JOB_FIELD_TYPE = 0x00000002,
  FAX_JOB_FIELD_QUEUE_STATUS = 0x00000004,
  FAX_JOB_FIELD_STATUS_EX = 0x00000008,
  FAX_JOB_FIELD_SIZE = 0x00000010,
  FAX_JOB_FIELD_PAGE_COUNT = 0x00000020,
  FAX_JOB_FIELD_CURRENT_PAGE = 0x00000040,
  FAX_JOB_FIELD_RECIPIENT_PROFILE = 0x00000080,
  FAX_JOB_FIELD_SCHEDULE_TIME = 0x00000100,
```

```
  FAX_JOB_FIELD_ORIGINAL_SCHEDULE_TIME = 0x00000200,
  FAX_JOB_FIELD_SUBMISSION_TIME = 0x00000400,
  FAX_JOB_FIELD_TRANSMISSION_START_TIME = 0x00000800,
  FAX_JOB_FIELD_TRANSMISSION_END_TIME = 0x00001000,
  FAX_JOB_FIELD_PRIORITY = 0x00002000,
  FAX_JOB_FIELD_RETRIES = 0x00004000,
  FAX_JOB_FIELD_DELIVERY_REPORT_TYPE = 0x00008000,
  FAX_JOB_FIELD_SENDER_PROFILE = 0x00010000,
  FAX_JOB_FIELD_STATUS_SUB_STRUCT = 0x00020000,
  FAX_JOB_FIELD_DEVICE_ID  = 0x00040000,
  FAX_JOB_FIELD_MESSAGE_ID = 0x00080000,
  FAX_JOB_FIELD_BROADCAST_ID = 0x00100000,
  FAX_JOB_FIELD_RECEIPT_TYPE = 0x00200000,
  FAX_JOB_FIELD_SERVER_RECEIVE_FOLDER = 0x00400000,
  FAX_JOB_FIELD_MESSAGE_FLAGS = 0x00800000
} FAX_ENUM_JOB_FIELDS;
```

FAX_JOB_FIELD_JOB_ID: The presence of this flag indicates that the **job ID** field is valid.

FAX_JOB_FIELD_TYPE: The presence of this flag indicates that the **job type** field is valid.

FAX_JOB_FIELD_QUEUE_STATUS: The presence of this flag indicates that the **queue status** field is valid.

FAX_JOB_FIELD_STATUS_EX: The presence of this flag indicates that the **extended status** field is valid.

FAX_JOB_FIELD_SIZE: The presence of this flag indicates that the **size** field is valid.

FAX_JOB_FIELD_PAGE_COUNT: The presence of this flag indicates that the **page count** field is valid.

FAX_JOB_FIELD_CURRENT_PAGE: The presence of this flag indicates that the **current page** field is valid.

FAX_JOB_FIELD_RECIPIENT_PROFILE: The presence of this flag indicates that the **recipient profile** field is valid.

FAX_JOB_FIELD_SCHEDULE_TIME: The presence of this flag indicates that the **schedule time** field is valid.

FAX_JOB_FIELD_ORIGINAL_SCHEDULE_TIME: The presence of this flag indicates that the **original schedule time** field is valid.

FAX_JOB_FIELD_SUBMISSION_TIME: The presence of this flag indicates that the **submission time** field is valid.

FAX_JOB_FIELD_TRANSMISSION_START_TIME: The presence of this flag indicates that the **transmission start time** field is valid.

FAX_JOB_FIELD_TRANSMISSION_END_TIME: The presence of this flag indicates that the **transmission end time** field is valid.

FAX_JOB_FIELD_PRIORITY: The presence of this flag indicates that the **priority** field is valid.

FAX_JOB_FIELD_RETRIES: The presence of this flag indicates that the **retries** field is valid.

FAX_JOB_FIELD_DELIVERY_REPORT_TYPE: The presence of this flag indicates that the **delivery report** field is valid.

FAX_JOB_FIELD_SENDER_PROFILE: The presence of this flag indicates that the **sender profile** field is valid.

FAX_JOB_FIELD_STATUS_SUB_STRUCT: The presence of this flag indicates that the **status** field is valid.

FAX_JOB_FIELD_DEVICE_ID : The presence of this flag indicates that the **device id** field is valid.

FAX_JOB_FIELD_MESSAGE_ID: The presence of this flag indicates that the **message id** field is valid.

FAX_JOB_FIELD_BROADCAST_ID: The presence of this flag indicates that the **broadcast id** field is valid.

FAX_JOB_FIELD_RECEIPT_TYPE: The presence of this flag indicates that the **receipt type** field is valid.

FAX_JOB_FIELD_SERVER_RECEIVE_FOLDER: The presence of this flag indicates that the **server receive folder** field is valid.

FAX_JOB_FIELD_MESSAGE_FLAGS: The presence of this flag indicates that the **message flag** field is valid.

2.2.64 FAX_ENUM_COVERPAGE_FORMATS

The **FAX_ENUM_COVERPAGE_FORMAT** enumeration defines the types of cover page which the server MUST support. The cover page MUST be one of the following values:

```
typedef  enum
{
  FAX_COVERPAGE_FMT_COV = 0x00000001,
  FAX_COVERPAGE_FMT_COV_SUBJECT_ONLY = 0x00000002
} FAX_ENUM_COVERPAGE_FORMATS;
```

FAX_COVERPAGE_FMT_COV: Indicates it is a normal cover page.

FAX_COVERPAGE_FMT_COV_SUBJECT_ONLY: Indicates it is a subject only cover page.

2.2.65 FAX_SPECIFIC_ACCESS_RIGHTS_2

The **FAX_SPECIFIC_ACCESS_RIGHTS_2** enumeration defines specific access rights, which provide security when users query and manage fax jobs, fax devices, and fax document. The access rights specified below define access rights in addition to those specified in **FAX_SPECIFIC_ACCESS_RIGHTS (section 2.2.16)**.

```
typedef  enum
{
  FAX ACCESS QUERY OUT JOBS = 0x0008,
  FAX ACCESS MANAGE OUT JOBS = 0x0010,
  FAX ACCESS QUERY ARCHIVES = 0x0080,
  FAX_ACCESS_MANAGE_ARCHIVES = 0x0100,
  FAX_ACCESS_MANAGE_RECEIVE_FOLDER = 0x0200,
```

```
  FAX_GENERIC_READ_2 = FAX_ACCESS_QUERY_CONFIG | FAX_ACCESS_MANAGE_RECEIVE_FOLDER,
  FAX_GENERIC_WRITE_2 = FAX_ACCESS_MANAGE_CONFIG,
  FAX_GENERIC_EXECUTE_2 = FAX_ACCESS_SUBMIT,
  FAX_GENERIC_ALL_2 = FAX_ACCESS_SUBMIT          | FAX_ACCESS_SUBMIT_NORMAL |
         FAX_ACCESS_SUBMIT_HIGH     | FAX_ACCESS_QUERY_OUT_JOBS |
                      FAX_ACCESS_MANAGE_OUT_JOBS | FAX_ACCESS_QUERY_CONFIG |
                      FAX_ACCESS_MANAGE_CONFIG   | FAX_ACCESS_QUERY_ARCHIVES |
                      FAX_ACCESS_MANAGE_ARCHIVES
} FAX_SPECIFIC_ACCESS_RIGHTS_2;
```

FAX_ACCESS_QUERY_OUT_JOBS: The user MAY view all the outgoing jobs in the server's queue.

FAX_ACCESS_MANAGE_OUT_JOBS: The user MAY manage all the outgoing jobs in the server's queue.

FAX_ACCESS_QUERY_ARCHIVES: The user MAY view all the messages (Inbox and Sent Items) in the server's archive.

FAX_ACCESS_MANAGE_ARCHIVES: The user MAY manage all the messages (Inbox and Sent Items) in the server's archive.

FAX_ACCESS_MANAGE_RECEIVE_FOLDER: The user MAY view and manage the server's incoming queue.

FAX_GENERIC_READ_2: Access rights needed to read faxes.

FAX_GENERIC_WRITE_2: Access rights needed to write faxes.

FAX_GENERIC_EXECUTE_2: Access rights needed to execute faxes.

FAX_GENERIC_ALL_2: All access rights.

2.2.66 FAX_EVENT_JOB

The **FAX_EVENT_JOB** structure defines information about notifications regarding a single job in the server's queue.

```
typedef struct {
  DWORDLONG dwlMessageId;
  FAX_ENUM Type;
  PFAX_JOB_STATUS pJobData;
} FAX_EVENT_JOB;
```

dwlMessageId: A DWORDLONG value that contains the unique identifier of the job.

Type: Specifies the type of information about the job. This will be one of the **FAX_ENUM_JOB_EVENT_TYPE (section 2.2.57)** enumeration values.

pJobData: If the **Type** member contains the FAX_JOB_EVENT_TYPE_STATUS value from the **FAX_ENUM_JOB_EVENT_TYPE** (section 2.2.57) enumeration, this member contains a pointer to a **FAX_JOB_STATUS** structure that contains the current status of the job. Otherwise, this member is NULL.

3 Protocol Details

The client side of this protocol is simply a pass-through. That is, there are no additional timers or other states required on the client side of this protocol. Calls made by the higher-layer protocol or application are passed directly to the transport, and the results returned by the transport are passed directly back to the higher-layer protocol or application.

3.1 Fax Server Details

3.1.1 Abstract Data Model

No abstract data model is required.

3.1.2 Timers

No protocol timer events are required on the client beyond the timers required in the underlying RPC protocol.

3.1.3 Initialization

The server MUST listen on a well-known endpoint, as specified in [C706].

3.1.4 Message Processing Events and Sequencing Rules

This protocol MUST indicate to the RPC runtime that it is to perform a strict NDR data consistency check at target level 5.0, as specified in [MS-RPCE] section 3.

This protocol MUST indicate to the RPC runtime via the **type_strict_context_handle** attribute that it is to reject use of context handles created by a method that creates a different type of context handle, as specified in [MS-RPCE] section 3.

Methods in RPC Opnum Order

Method	Description
FAX_GetServicePrinters	Opnum: 0
FAX_ConnectionRefCount	The server connects or disconnects the fax client to or from the fax service. Opnum: 1
FAX_OpenPort	The server opens a fax port for subsequent use in other fax methods and returns a fax porthandle for use by the fax client application. Opnum: 2
FAX_ClosePort	This method is called by the client to close an open fax port. Opnum: 3
FAX_EnumJobs	This method is called by the client to enumerate all the fax jobs on the specified fax server. Opnum: 4
FAX_GetJob	The **FAX_GetJob** method is called by the client to retrieve information regarding a specific job.

Method	Description
	Opnum: 5
FAX_SetJob	The **FAX_SetJob** method is called by the client. In response, the server pauses, resumes, cancels, or restarts the specified fax job. Opnum: 6
FAX_GetPageData	This function is called by the client to retrieve data in the first page of an outgoing fax job. Opnum: 7
FAX_GetDeviceStatus	The **FAX_GetDeviceStatus** method is called by the client to retrieve information regarding a specified fax device (port). Opnum: 8
FAX_Abort	The **FAX_Abort** method is called by the client to abort the specified fax job on the server. Opnum: 9
FAX_EnumPorts	The **FAX_EnumPorts** method is called by the client to obtain port state information. Opnum: 10
FAX_GetPort	This function is called by the client to retrieve port status information for a requested port at the server. Opnum: 11
FAX_SetPort	A fax client application uses this function to set fax device information. Opnum: 12
FAX_EnumRoutingMethods	The **FAX_EnumRoutingMethods** method is called by the client to enumerate all the routing methods that are registered with the specified fax server. Opnum: 13
FAX_EnableRoutingMethod	The **FAX_EnableRoutingMethod** method is called by the client for a specified fax device (port). Opnum: 14
FAX_GetRoutingInfo	This function is called by the client to retrieve information regarding a specified routing method identified by the passed in GUID. Opnum: 15
FAX_SetRoutingInfo	The **FAX_SetRoutingInfo** method is called by the client to set routing information for a fax routing method. Opnum: 16
FAX_EnumGlobalRoutingInfo	The **FAX_EnumGlobalRoutingInfo** method is called by the client to enumerate global routing information. Opnum: 17
FAX_SetGlobalRoutingInfo	The fax client application calls this function to set global

Method	Description
	routing properties like the routing method priority. Opnum: 18
FAX_GetConfiguration	The **FAX_GetConfiguration** method is called by the client. In response, the server MUST validate that the client has access to query configuration. It then MUST allocate memory for the configuration information to be passed out and then fill it with data. Opnum: 19
FAX_SetConfiguration	The fax client application calls this function to change the Fax configuration on the Fax server. The SizeOfStruct in the FaxConfig argument MUST be set to a value of sizeof(FAX_CONFIGURATIONW). Opnum: 20
FAX_GetLoggingCategories	The **FAX_GetLoggingCategories** method is called by the client. In response, the server MUST return the current logging categories for the fax server to which the client has connected. Opnum: 21
FAX_SetLoggingCategories	The **FAX_SetLoggingCategories** method is called by the client. In response, the server modifies the current logging categories for the fax server to which the client has connected. Opnum: 22
FAX_GetSecurity	The FAX_GetSecurity method was called by the client to the retrieve information regarding the FAX security descriptor from the FAX server. This function is no longer supported. **FAX_GetSecurityEx2** MUST be used instead of this function Opnum: 23
FAX_SetSecurity	The **FAX_SetSecurity** method is called by the client. In response, the server sets the fax server's security descriptor. Opnum: 24
FAX_AccessCheck	The **FAX_AccessCheck** method is called when the client wants to check if it has access permissions to do a particular server operation. Opnum: 25
FAX_CheckServerProtSeq	The **FAX_CheckServerProtSeq** method is called by the client. In response, the server should return 50 (ERROR_NOT_SUPPORTED). Opnum: 26
FAX_SendDocumentEx	The **FAX_SendDocumentEx** method is called by the client. In response, the server sends the specified fax job. Opnum: 27
FAX_EnumJobsEx	The **FAX_EnumJobsEx** method is called by the client to enumerate a specified set of jobs on the server's queue.

Method	Description
	Opnum: 28
FAX_GetJobEx	The **FAX_GetJobEx** is called by the client to retrieve information regarding a specified job at the server. Opnum: 29
FAX_GetCountryList	The **FAX_GetCountryList** method is called by the client to retrieve the list of country/region information defined on the server. Opnum: 30
FAX_GetPersonalProfileInfo	The **FAX_GetPersonalProfileInfo** method is called by the client to retrieve information on the personal profile of a user from the specified fax message present in the described message folder. Opnum: 31
FAX_GetQueueStates	The **FAX_GetQueueStates** method is called by the client to retrieve the state of the fax outgoing queue at the server. Opnum: 32
FAX_SetQueue	The fax client calls the **FAX_SetQueue** method to change the state of the server queue. Opnum: 33
FAX_GetReceiptsConfiguration	The **FAX_GetReceiptsConfiguration** method is called by the client. In response, the server returns the receipts configuration information of the fax server. Opnum: 34
FAX_SetReceiptsConfiguration	The **FAX_SetReceiptsConfiguration** method is called by the client. In response, the server sets the receipts configuration information used by the fax server to send delivery receipts for fax transmissions. Opnum: 35
FAX_GetReceiptsOptions	The **FAX_GetReceiptsOptions** method is called by the client to the retrieve the supported receipt options on the server. Opnum: 36
FAX_GetVersion	The fax client application calls the **FAX_GetVersion** method to get the version of the fax server it is connected to. Opnum: 37
FAX_GetOutboxConfiguration	The **FAX_GetOutboxConfiguration** method is called by the client to retrieve the outbox configuration at the server. Opnum: 38
FAX_SetOutboxConfiguration	The fax client application calls the **FAX_SetOutboxConfiguration** method to set the current Outbox configuration, such as the Discount Time. Opnum: 39
FAX_GetPersonalCoverPagesOption	The **FAX_GetPersonalCoverPagesOption** method is called by the client to retrieve information about the supported

Method	Description
	personal cover-page options. Opnum: 40
FAX_GetArchiveConfiguration	The **FAX_GetArchiveConfiguration** method is called by the client to retrieve the current archive configuration on the fax server. Opnum: 41
FAX_SetArchiveConfiguration	The fax client application calls the **FAX_SetArchiveConfiguration** method to set the archive configuration for a specific fax folder on the fax server. Opnum: 42
FAX_GetActivityLoggingConfiguration	he **FAX_GetActivityLoggingConfiguration** method is called by the client to retrieve the current activity logging configuration. Opnum: 43
FAX_SetActivityLoggingConfiguration	The fax client application calls the **FAX_SetActivityLoggingConfiguration** method to set options for activity logging. Opnum: 44
FAX_EnumerateProviders	The **FAX_EnumerateProviders** method is called by the client to enumerate all the fax service providers (FSPs) that are installed on the server. Opnum: 45
FAX_GetPortEx	The **FAX_GetPortEx** method is called by the client to retrieve port status information for a requested port at the server. Opnum: 46
FAX_SetPortEx	A fax client application uses the **FAX_SetPortEx** method to set fax device information. Opnum: 47
FAX_EnumPortsEx	The **FAX_EnumPortsEx** method is called by the client to enumerate detailed port state information for each device connected to the fax server. Opnum: 48
FAX_GetExtensionData	The **FAX_GetExtensionData** method is called by the client to retrieve a fax extension's private data. Opnum: 49
FAX_SetExtensionData	The fax client application calls the **FAX_SetExtensionData** method in order to write the extension data for a device. Opnum: 50
FAX_AddOutboundGroup	The **FAX_AddOutboundGroup** method is called by the client to request a new outbound routing group. Opnum: 51

Method	Description
FAX_SetOutboundGroup	The fax client application calls the **FAX_SetOutboundGroup** method to set a new device list to an existing group. Opnum: 52
FAX_RemoveOutboundGroup	The fax client application calls the **FAX_RemoveOutboundGroup** method to remove an existing outbound routing group from the fax server. Opnum: 53
FAX_EnumOutboundGroups	The **FAX_EnumOutboundGroups** method is called by the client. In response, the server MUST validate the client has access to enumerate the outbound routing groups. Opnum: 54
FAX_SetDeviceOrderInGroup	The **FAX_SetDeviceOrderInGroup** method is called by the client. In response, the server sets the order of a single device in a group of outbound routing devices. Opnum: 55
FAX_AddOutboundRule	The **FAX_AddOutboundRule** method is called by the client to request a new outbound routing rule in the specified routing group. Opnum: 56
FAX_RemoveOutboundRule	The **FAX_RemoveOutboundRule** method removes an existing outbound routing rule from the rules map. Opnum: 57
FAX_SetOutboundRule	A fax client that the client has access to submit either normal or high priority faxes. Opnum: 58
FAX_EnumOutboundRules	The **FAX_EnumOutboundRules** method is called by the client to enumerate all the outbound routing rules that are present on the specified fax server. Opnum: 59
FAX_RegisterServiceProviderEx	The fax client application calls the **FAX_RegisterServiceProviderEx** method to register a fax service provider (FSP) with the fax service. Opnum: 60
FAX_UnregisterServiceProviderEx	The **FAX_UnregisterServiceProviderEx** method is called when the client wants to unregister a fax service provider (FSP). Opnum: 61
FAX_UnregisterRoutingExtension	The **FAX_UnregisterRoutingExtension** method unregisters an existing inbound routing extension. For unregistration to occur, the fax server MUST be restarted. Opnum: 62
FAX_StartMessagesEnum	The **FAX_StartMessagesEnum** method is called by the client. In response, the server starts enumerating messages

Method	Description
	in one of the archives. Opnum: 63
FAX_EndMessagesEnum	The **FAX_EndMessagesEnum** method is called by the client. On success, the server MUST halt the enumerating messages in the specified archives. Opnum: 64
FAX_EnumMessages	The **FAX_EnumMessages** method is called by the client. In response, the server MUST validate that the hEnum argument passed by the client was created as part of a prior **FAX_StartMessagesEnum** call. Opnum: 65
FAX_GetMessage	The **FAX_GetMessage** method is called by the client. The archive can be one of the enumerations defined by FAX_ENUM_MESSAGE_FOLDER except FAX_MESSAGE_FOLDER_QUEUE. Opnum: 66
FAX_RemoveMessage	The fax client application calls the **FAX_RemoveMessage** method to remove a message from a specific fax archive folder. Opnum: 67
FAX_StartCopyToServer	The client calls the **FAX_StartCopyToServer** method to start a copy of a file to the server queue that the client has access to submit either normal or high priority faxes. Opnum: 68
FAX_StartCopyMessageFromServer	The **FAX_StartCopyMessageFromServer** method starts a copy process of a message from the server's archive or queue to the caller. Opnum: 69
FAX_WriteFile	The **FAX_WriteFile** method is called by the client. In response, the server receives a file and places it in the queue. Opnum: 70
FAX_ReadFile	The fax client application calls the **FAX_ReadFile** method to copy a file from the server (in chunks). Opnum: 71
FAX_EndCopy	The **FAX_EndCopy** method is called by the client to end a copy process from or to the server. Opnum: 72
FAX_StartServerNotification	The **FAX_StartServerNotification** method is called by the client. In response, the server should return 50 (ERROR_NOT_SUPPORTED). Opnum: 73
FAX_StartServerNotificationEx	The **FAX_StartServerNotificationEx** method is called by the client. In response, the server starts the fax notification

Method	Description
	server. Opnum: 74
FAX_EndServerNotification	The **FAX_EndServerNotification** method is called by the client to stop the notifications from the server. Opnum: 75
FAX_GetServerActivity	The fax client application calls the **FAX_GetServerActivity** method to retrieve the status of the fax server queue activity and event log reports. Opnum: 76
FAX_SetConfigWizardUsed	The **FAX_SetConfigWizardUsed** method is called by the client. In response, the server sets a value in the registry indicating that the configuration wizard was used. Opnum: 77
FAX_EnumRoutingExtensions	The **FAX_EnumRoutingExtensions** function is called by the client to enumerate all the routing extensions that are registered with the specified fax server. Opnum: 78
FAX_AnswerCall	The **FAX_AnswerCall** method is called by the client to cause the server to answer the specified call. Opnum: 79
FAX_ConnectFaxServer	The **FAX_ConnectFaxServer** method is called by the client to create an initial connection to the server. Opnum: 80
FAX_GetSecurityEx	This function is called by the client to retrieve information about the fax security descriptor from the fax server. Opnum: 81
FAX_RefreshArchive	A fax client application calls the **FAX_RefreshArchive** to notify the server that the archive folder has changed and should be refreshed. Opnum: 82
FAX_SetRecipientsLimit	A fax client application calls the **FAX_SetRecipientsLimit** to set the recipients limit of a single broadcast job. Opnum: 83
FAX_GetRecipientsLimit	This function is called by the client to the retrieve information regarding the recipients limit of a single broadcast job. Opnum: 84
FAX_GetServerSKU	The **FAX_GetServerSKU** method is called by the client. In response, the server returns the Stock-Keeping Unit (SKU) of the fax server operating system. Opnum: 85
FAX_CheckValidFaxFolder	The **FAX_CheckValidFaxFolder** method is called by the client to check if the specified path is accessible to the fax

Method	Description
	server. Opnum: 86
FAX_GetJobEx2	The **FAX_GetJobEx2** method is called by the client to retrieve information regarding a specified job. Opnum: 87
FAX_EnumJobsEx2	The **FAX_EnumJobsEx2** method is called by the client to enumerate a specified set of jobs on the server's queue for a specific fax account. Opnum: 88
FAX_GetMessageEx	This function is called by the client to retrieve a particular message from one of the specified fax message archives. Opnum: 89
FAX_StartMessagesEnumEx	The **FAX_StartMessagesEnumEx** method is called by the client. In response, the server starts enumerating messages in one of the archives. Opnum: 90
FAX_EnumMessagesEx	The **FAX_EnumMessagesEx** method is called by the client. This message differs from the **FAX_EnumMessages** method in that this function takes a level parameter, which differentiates the type of the message information structure the function returns. Opnum: 91
FAX_StartServerNotificationEx2	The **FAX_StartServerNotificationEx2** method is called by the client. In response, the server starts the fax notification server. Opnum: 92
FAX_CreateAccount	The **FAX_CreateAccount** method is called by the client to request a new fax account. Opnum: 93
FAX_DeleteAccount	The **FAX_DeleteAccount** method is called by the client. On success, the server MUST delete the specified fax account. The server MUST validate that the user has access to delete accounts. Opnum: 94
FAX_EnumAccounts	The **FAX_EnumAccounts** method is called by the client to enumerate all the fax accounts on the server. Opnum: 95
FAX_GetAccountInfo	The **FAX_GetAccountInfo** method is called by the client to retrieve information about a specified account. Opnum: 96
FAX_GetGeneralConfiguration	The **FAX_GetGeneralConfiguration** method is called by the client to request information regarding the general configuration at the server.

Method	Description
	Opnum: 97
FAX_SetGeneralConfiguration	The fax client application calls this function to set the configuration options provided for the fax service. Opnum: 98
FAX_GetSecurityEx2	This function is called by the client to retrieve information about the fax security descriptor from the fax server. Opnum: 99
FAX_SetSecurityEx2	The **FAX_SetSecurityEx2** method is called by the client. In response, the server sets the security descriptor of the fax server. Opnum: 100
FAX_AccessCheckEx2	The **FAX_AccessCheckEx2** method is called by the client when the client wants to check if it has access permissions for a particular server operation. Opnum: 101
FAX_ReAssignMessage	The fax client application calls this function to assign the specified fax message to a set of users. Opnum: 102
FAX_SetMessage	The fax client application calls this function to set the specific message properties for the message that is identified by its ID. Opnum: 103
FAX_GetConfigOption	The **FAX_GetConfigOption** is called by the client to retrieve a configuration setting at the server using an RPC_REQUEST packet. Opnum: 104

All methods MUST NOT throw exceptions.

3.1.4.1 FAX_Abort (Opnum 9)

The **FAX_Abort (Opnum 9)** method is called by the client to abort the specified fax job on the server.

In response, the server MUST validate that the job ID is for a valid job. The server MUST validate that the client has write access to the job. On success, the server MUST terminate the specified fax job that is queued or in progress.

```
error_status_t FAX_Abort(
  [in] handle_t hBinding,
  [in] DWORD JobId
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

JobId: A unique number that identifies the fax job to terminate.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x000010DD ERROR_INVALID_OPERATION	The operation identifier is not valid.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.2 FAX_AccessCheck (Opnum 25)

The **FAX_AccessCheck (Opnum 25)** method is called when the client wants to check if it has access permissions to do a particular server operation.

In response, the server MUST validate the desired access right against the specific access rights. On success, the server SHOULD return the access rights.

```
error_status_t FAX_AccessCheck(
  [in] handle_t hBinding,
  [in] DWORD AccessMask,
  [out, ref] BOOL* pfAccess,
  [in, out, unique] LPDWORD lpdwRights
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

AccessMask: A **DWORD** variable that MUST contain a set of bit flags that define the fax access permissions for a user. This parameter can be any combination of fax-specific access rights, standard access rights, and generic access rights. For a list of standard access rights, see [MSDN-SAR].

Generic Access Rights	Meaning
FAX_GENERIC_READ 0x000002A8	Includes the read-only rights that are granted by the following specific access rights: ▪ FAX_ACCESS_QUERY_JOBS

Generic Access Rights	Meaning
	▪ FAX_ACCESS_QUERY_CONFIG ▪ FAX_ACCESS_QUERY_IN_ARCHIVE ▪ FAX_ACCESS_QUERY_OUT_ARCHIVE
FAX_GENERIC_WRITE 0x00000550	Includes the management rights that are granted by the following specific access rights: ▪ FAX_ACCESS_MANAGE_JOBS ▪ FAX_ACCESS_MANAGE_CONFIG ▪ FAX_ACCESS_MANAGE_IN_ARCHIVE ▪ FAX_ACCESS_MANAGE_OUT_ARCHIVE
FAX_GENERIC_EXECUTE 0x00000001	Identical to the FAX_ACCESS_SUBMIT access right.
FAX_GENERIC_ALL 0x000007FF	Includes all the following specific fax permissions: ▪ FAX_ACCESS_SUBMIT ▪ FAX_ACCESS_SUBMIT_NORMAL ▪ FAX_ACCESS_SUBMIT_HIGH ▪ FAX_ACCESS_QUERY_JOBS ▪ FAX_ACCESS_MANAGE_JOBS ▪ FAX_ACCESS_QUERY_CONFIG ▪ FAX_ACCESS_MANAGE_CONFIG ▪ FAX_ACCESS_QUERY_IN_ARCHIVE ▪ FAX_ACCESS_MANAGE_IN_ARCHIVE ▪ FAX_ACCESS_QUERY_OUT_ARCHIVE ▪ FAX_ACCESS_MANAGE_OUT_ARCHIVE

Specific Access Rights	Meaning
FAX_ACCESS_SUBMIT 0x00000001	Grants permission to send a low priority fax transmission to one or more recipients.
FAX_ACCESS_SUBMIT_NORMAL 0x00000002	Grants permission to send a normal priority fax transmission to one or more recipients.
FAX_ACCESS_SUBMIT_HIGH	Grants permission to send a high priority fax transmission

Specific Access Rights	Meaning
0x00000004	to one or more recipients.
FAX_ACCESS_QUERY_JOBS 0x00000008	Grants permission to view all the incoming and outgoing faxes in the Incoming and Outbox queues, including those that belong to other users. By default, without this permission, non-administrator users can view their own outgoing messages in the Outbox queue, but cannot view the Incoming queue. Also, non-administrator users cannot view incoming or outgoing faxes that belong to other users.
FAX_ACCESS_MANAGE_JOBS 0x00000010	Grants permission to manage all the incoming and outgoing faxes in the Incoming and Outbox queues, including those that belong to other users. By default, without this permission, non-administrator users can manage their own outgoing messages in the Outbox queue, but cannot manage the Incoming queue. Also, non-administrator users cannot manage incoming or outgoing faxes that belong to other users.
FAX_ACCESS_QUERY_CONFIG 0x00000020	Grants permission to view the properties of the Fax Service. By default, non-administrator users do not have this permission. Without this permission, users cannot view any of the tree nodes, except for the cover page node in the Fax Service Manager.
FAX_ACCESS_MANAGE_CONFIG 0x00000040	Grants permission to modify the properties of the fax service. By default, non-administrator users do not have this permission.
FAX_ACCESS_QUERY_IN_ARCHIVE 0x00000080	Grants permission to view all successfully received messages in the Inbox archive. By default, without this permission, non-administrator users cannot view archived incoming faxes.
FAX_ACCESS_MANAGE_IN_ARCHIVE 0x00000100	Grants permission to manage all successfully received messages in the Inbox archive. By default, without this permission, non-administrator users cannot manage archived incoming faxes.
FAX_ACCESS_QUERY_OUT_ARCHIVE 0x00000200	Grants permission to view all successfully sent messages in the Sent Items archive, including those belonging to other users. By default, without this permission, non-administrator users can view archives of their own sent messages but cannot view archives that belong to other users.
FAX_ACCESS_MANAGE_OUT_ARCHIVE 0x00000400	Grants permission to manage all successfully sent messages in the Sent Items archive, including those that belong to other users. By default, without this permission, non-administrator users can manage archives of their own sent messages but cannot manage archives that belong to other users.

Standard Access Rights	Meaning
DELETE 0x00000010	Delete access.
READ_CONTROL 0x00000011	Read access to the owner, group, and discretionary access control list (ACL) of the security descriptor.
WRITE_DAC 0x00000012	Write access to the ACL.
WRITE_OWNER 0x00000013	Write access to the owner.
SYNCHRONIZE 0x00000014	Windows NT synchronize access.

Miscellaneous Access Rights	Meaning
MAXIMUM_ALLOWED 0x02000000	Maximum allowed access rights for this server.

pfAccess: A pointer to a **BOOL** to receive the access check return value. **TRUE** indicates that access is allowed.

lpdwRights: A pointer to a **DWORD** value to receive the access rights as a bitwise OR. A client can request that the maximum allowed rights are granted by setting the *dwAccessMask* parameter to 0x02000000 (MAXIMUM_ALLOWED). The server should set the out value to the actual rights granted.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.3 FAX_AccessCheckEx2 (Opnum 101)

The **FAX_AccessCheckEx2 (Opnum 101)** method is called by the client when the client wants to check if it has access permissions for a particular server operation.

In response the server MUST validate the desired access right against the fax service security descriptor. On success, the server SHOULD return the access rights.<6>

```
error_status_t FAX_AccessCheckEx2(
  [in] handle_t hBinding,
  [in] DWORD AccessMask,
  [out, ref] BOOL* pfAccess,
  [in, out, unique] LPDWORD lpdwRights
);
```

hBinding: The fax server handle that is returned by a call to the **Fax_ConnectFaxServer (Opnum 80)** method. This is a context handle

AccessMask: A **DWORD** variable that contains a set of bit flags defining a user's fax access permissions. This parameter can be any combination of fax-specific access rights, standard access rights, and generic access rights.

Generic access rights	Meaning
FAX_GENERIC_EXECUTE_2 0x00000001	Includes the read-only rights granted by the FAX_ACCESS_SUBMIT access right.
FAX_GENERIC_READ_2 0x00000020	Includes the read-only rights granted by the FAX_ACCESS_QUERY_CONFIG access right.
FAX_GENERIC_WRITE_2 0x00000040	Includes the read-only rights granted by the FAX_ACCESS_MANAGE_CONFIG access right.
FAX_GENERIC_ALL_2 0x000003FF	Includes the read-only rights granted by the following specific access rights: ▪ FAX_ACCESS_SUBMIT ▪ FAX_ACCESS_SUBMIT_NORMAL ▪ FAX_ACCESS_SUBMIT_HIGH ▪ FAX_ACCESS_QUERY_OUT_JOBS ▪ FAX_ACCESS_MANAGE_OUT_JOBS ▪ FAX_ACCESS_QUERY_CONFIG ▪ FAX_ACCESS_MANAGE_CONFIG ▪ FAX_ACCESS_QUERY_ARCHIVES ▪ FAX_ACCESS_MANAGE_ARCHIVES ▪ FAX_ACCESS_MANAGE_RECEIVE_FOLDER

Specific Access Rights	Meaning
FAX_ACCESS_SUBMIT	Grants permission to send a low-priority fax

Specific Access Rights	Meaning
0x00000001	transmission to one or more recipients.
FAX_ACCESS_SUBMIT_NORMAL 0x00000002	Grants permission to send a normal-priority fax transmission to one or more recipients.
FAX_ACCESS_SUBMIT_HIGH 0x00000004	Grants permission to send a high-priority fax transmission to one or more recipients.
FAX_ACCESS_QUERY_OUT_JOBS 0x00000008	Grants permission to view the outgoing faxes in the fax queue. By default, no users have this permission.
FAX_ACCESS_MANAGE_OUT_JOBS 0x00000010	Grants permission to manage the outgoing faxes in the fax queue by using such operations as pause, resume, restart, and delete (**FAX_SetJob**). By default, no users have this permission.
FAX_ACCESS_QUERY_CONFIG 0x00000020	Grants permission to view the properties of the Fax Service and enumerate accounts, and to read any account configuration information. By default, non-administrator users do not have this permission. Without it, users cannot view any of the tree nodes, except for the cover page node in the Fax Service Manager.
FAX_ACCESS_MANAGE_CONFIG 0x00000040	Grants permission to modify the properties of the fax service. By default, non-administrator users do not have this permission.
FAX_ACCESS_QUERY_ARCHIVES 0x00000080	Grants permission to view the sent and received fax messages in the archives. By default, no users have this permission.
FAX_ACCESS_MANAGE_ARCHIVES 0x00000100	Grants permission to manage the sent and received fax messages in the archives by using such operations as delete (**FAX_RemoveMessage**) and copy (**FAX_StartCopyMessageFromServer**, **FAX_StartCopyToServer**, **FAX_EndCopy**). By default, no users have this permission.
FAX_ACCESS_MANAGE_RECEIVE_FOLDER 0x00000200	When global routing is not enabled, this permission allows the user to delete any messages. When global routing is active, it allows the user to see the contents of all receive folder faxes, to delete faxes, and to cancel receive transmissions in progress.

pfAccess: A pointer to a Boolean value that receives the access check return value. If this value is **TRUE**, access is allowed.

lpdwRights: A pointer to a **DWORD** value that receives the access rights as a bitwise OR operation. A client can request that the maximum allowed rights are granted by setting the *dwAccessMask* parameter to 0x02000000 (MAXIMUM_ALLOWED). The server should set the out value to the actual rights granted.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.4 FAX_AddOutboundGroup (Opnum 51)

The **FAX_AddOutboundGroup (Opnum 51)** method is called by the client to request a new outbound routing group.

In response, the server MUST check for access to write outbound groups. The server MUST check for duplicate group names in a case-insensitive manner and if found, return ERROR_DUP_NAME. On success, the server MUST add a new outbound routing group to the fax server.

```
error_status_t FAX_AddOutboundGroup(
  [in] handle_t hFaxHandle,
  [in, string, ref] LPCWSTR lpwstrGroupName
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpwstrGroupName: A pointer to a null-terminated string that uniquely identifies a new group name. This value cannot be **NULL**. The group name is expected to be case-insensitive.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000034 ERROR_DUP_NAME	You were not connected because a duplicate name exists on the network. Go to System in Control Panel to change the computer name and try again.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.
0x000003F7	The registry is corrupted. The structure of one of the

Return value/code	Description
ERROR_REGISTRY_CORRUPT	files that contain registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.
0x00001B63 FAX_ERR_NOT_SUPPORTED_ON_THIS_SKU	The fax server API version does not support the requested operation.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.5 FAX_AddOutboundRule (Opnum 56)

The **FAX_AddOutboundRule (Opnum 56)** method is called by the client to request a new outbound routing rule in the specified routing group.

In response, if bUseGroup is **TRUE**, the server MUST validate that the group name is valid; if bUseGroup is **FALSE**, the server MUST validate that the device ID is for a valid device. The server MUST validate that the client has access to add an outbound routing rule.

On success, the server MUST add an outbound routing rule to the fax server.

```
error_status_t FAX_AddOutboundRule(
  [in] handle_t hFaxHandle,
  [in] DWORD dwAreaCode,
  [in] DWORD dwCountryCode,
  [in] DWORD dwDeviceId,
  [in, string, unique] LPCWSTR lpwstrGroupName,
  [in] BOOL bUseGroup
);
```

hFaxHandle: An RPC context handle that references a specified fax port.

dwAreaCode: The area code of the rule. If this value is zero, the special <All other areas> value SHOULD be used. The combination of the *dwAreaCode* and *dwCountryCode* parameters are a unique key.

dwCountryCode: The country/region code of the rule. If this value is zero, the special <All other countries> value SHOULD be used. The combination of the *dwAreaCode* and *dwCountryCode* parameters are a unique key.

dwDeviceId: The destination device of the rule. This value is valid only if the *bUseGroup* parameter is **FALSE**. A zero value is invalid.

lpwstrGroupName: The destination group of the rule. This value is valid only if the *bUseGroup* parameter is **TRUE**.

bUseGroup: A Boolean value that specifies whether the group should be used as the destination.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the

standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000014 ERROR_BAD_UNIT	The system cannot find the specified device.
0x0000001F ERROR_GEN_FAILURE	An attached system device is not functioning.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.
0x000003F7 ERROR_REGISTRY_CORRUPT	The registry is corrupted. The structure of one of the files containing registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.
0x00001B5B FAX_ERR_BAD_GROUP_CONFIGURATION	The fax server encountered an outbound routing group with bad configuration.
0x00001B63 FAX_ERR_NOT_SUPPORTED_ON_THIS_SKU	The fax server API version does not support the requested operation.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.6 FAX_AnswerCall (Opnum 79)

The **FAX_AnswerCall (Opnum 79)** method is called by the client to cause the server to answer the specified call.

In response the server MUST validate that the Device ID is for a valid device. The server MUST validate that the line exists and the call can be answered. The server MUST validate that the client has read access to the device. To indicate success, the server MUST immediately answer the incoming call on the specified line.<7>

```
error_status_t FAX_AnswerCall(
  [in] handle_t hFaxHandle,
  [in] DWORD dwDeviceId
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwDeviceId: A **DWORD** value indicating the fax server-generated line ID.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below:

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x000000AA ERROR_BUSY	The requested resource is in use.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.7 FAX_CheckServerProtSeq (Opnum 26)

The **FAX_CheckServerProtSeq (Opnum 26)** method is called by the client. In response, the server should return 50 (ERROR_NOT_SUPPORTED).<8>

```
error_status_t FAX_CheckServerProtSeq(
  [in] handle_t hBinding,
  [in, out, unique] LPDWORD lpdwProtSeq
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpdwProtSeq: A variable into which the requested sequence is specified. Upon return, *lpdwProtSeq* contains the value for the validated sequence.

Value	Meaning
RPC_PROT_NOT_SUPPORTED 0	The protocol sequence is not supported.
RPC_PROT_TCP_IP 1	Check the protocol sequence for TCP/IP.
RPC_PROT_SPX 2	Check the protocol sequence for IPX/SPX.

Return Values: This method always returns:

ERROR_NOT_SUPPORTED (0x00000032)

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

Note

This function is obsolete.

3.1.4.8 FAX_CheckValidFaxFolder (Opnum 86)

The **FAX_CheckValidFaxFolder (Opnum 86)** method is called by the client to check if the specified path is accessible to the fax server.

In response the server MUST validate the folder path. The server MUST validate that the folder has correct access rights and does not collide with the queue or Inbox folder. On success, the server MUST confirm that the specified path is accessible for use by the Fax Service.<9>

```
error_status_t FAX_CheckValidFaxFolder(
  [in] handle_t hBinding,
  [in, in, string, ref] LPCWSTR lpcwstrPath
);
```

hBinding: The handle that the client remote procedure call (RPC) layer provides when the RPC call is made.

lpcwstrPath: A pointer to a string that contains the path to validate.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00001B60 FAX_ERR_FILE_ACCESS_DENIED	The fax server cannot find the job or message by searching for its identifier.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.9 FAX_ClosePort (Opnum 3)

The **FAX_ClosePort (Opnum 3)** method is called by the client to close an open fax port. The client passes the FaxPortHandle, which it received from a call to **FAX_OpenPort (section 3.1.4.65)**.

In response, the server MUST validate the FaxPortHandle. On success, the server MUST close the specified port.

```
error_status_t FAX_ClosePort(
  [in, out] PRPC_FAX_PORT_HANDLE FaxPortHandle
);
```

FaxPortHandle: A pointer to a fax porthandle.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.10 FAX_ConnectFaxServer (Opnum 80)

The **FAX_ConnectFaxServer (Opnum 80)** method is called by the client to create an initial connection to the server.

In response, the server MUST validate whether the client has any fax user access rights. On success, the server MUST create an initial connection handle, which the client MUST use while making future function calls to the server.

```
error_status_t FAX_ConnectFaxServer(
  [in] handle_t hBinding,
  [in] DWORD dwClientAPIVersion,
  [out, ref] LPDWORD lpdwServerAPIVersion,
  [out, ref] PRPC_FAX_SVC_HANDLE pHandle
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwClientAPIVersion: A **DWORD** that MUST contain the API version of the client module. The value determines the specific FAX_ERR error codes that should be returned from the service.

Value	Meaning
FAX_API_VERSION_0 0x00000000	No FAX_ERR_* values are returned.

Value	Meaning
FAX_API_VERSION_1 0x00010000	FAX_ERR 7001-7012 can be returned from the server.
FAX_API_VERSION_2 0x00020000	FAX_ERR 7001-7013 can be returned from the server.
FAX_API_VERSION_3 0x00030000	FAX_ERR 7001-7013 can be returned from the server.

lpdwServerAPIVersion: A pointer to a **DWORD** that MUST contain the API version of the server module.

pHandle: The binding handle that is used by the RPC client to reference the RPC server where the call is directed. This parameter appears on the wire.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.11 FAX_ConnectionRefCount (Opnum 1)

The **FAX_ConnectionRefCount (Opnum 1)** method is called by the client.

In response, the server connects or disconnects the fax client to or from the Fax Service.

```
error_status_t FAX_ConnectionRefCount(
  [in] handle_t hBinding,
  [in, out] PRPC_FAX_SVC_HANDLE Handle,
  [in] DWORD Connect,
  [out] LPDWORD CanShare
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Handle: An RPC context handle that references a connection to the Fax Service. This parameter is used in other calls to the service.

Connect: A **DWORD** value that specifies connection information.

Value	Meaning
Disconnect 0	Close the Fax Service connection.
Connect 1	Connect to the Fax Service.
Release 2	Release a connection to the Fax Service.

CanShare: Always set to 1 for a connect call.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Fax clients call this method to connect or disconnect from the Fax Service.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.12 FAX_CreateAccount (Opnum 93)

The **FAX_CreateAccount (Opnum 93)** method is called by the client to request a new fax account.

The server MUST validate that the client has the access to create an account. On success, the server MUST create a new fax account, and return the fax account information in *Buffer*. The function MUST return failure if the account already exists.

The client SHOULD free the returned *Buffer*.<10>

```
error_status_t FAX_CreateAccount(
  [in] handle_t hBinding,
  [in] DWORD level,
  [in, ref, size_is(BufferSize)]
    const LPBYTE Buffer,
  [in, range (0,FAX_MAX_RPC_BUFFER)]
    DWORD BufferSize
);
```

hBinding: The fax server handle that is returned by a call to the **Fax_ConnectFaxServer (Opnum 80)** method. This is a context handle.

level: A **DWORD** value that indicates the type of structure to return in *Buffer*. The value passed in this parameter MUST be zero.

Buffer: A pointer to a **FAX_ACCOUNT_INFO_0 (section 2.2.18)** structure that contains fax account information.

BufferSize: A **DWORD** value that indicates the return size, in bytes, of the buffer that is pointed to by the *Buffer* parameter.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00000649 ERROR_INVALID_HANDLE	The handle is in an invalid state.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.13 FAX_DeleteAccount (Opnum 94)

The **FAX_DeleteAccount (Opnum 94)** method is called by the client.

On success, the server MUST delete the specified fax account. The server MUST validate that the user has access to delete accounts.<11>

```
error_status_t FAX_DeleteAccount(
  [in] handle_t hBinding,
  [in, string, unique] LPCWSTR lpcwstrAccountName
);
```

hBinding: The fax server handle that is returned by a call to the **Fax_ConnectFaxServer (Opnum 80)** method. This is a context handle.

lpcwstrAccountName: A pointer to a constant, null-terminated character string that contains the name of the account to delete.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the

standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00000649 ERROR_INVALID_HANDLE	The handle is in an invalid state.

The account name that *lpcwstrAccountName* indicates must be in one of the following formats. Any other format is invalid.

Format	Description
<machine_name>\<user_name>	For a local user with machine_name as the local machine's name.
<domain_name>\<user_name>	For a nonlocal user.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.14 FAX_EnableRoutingMethod (Opnum 14)

The **FAX_EnableRoutingMethod (Opnum 14)** method is called by the client for a specified fax device (port).

On success, the server MUST enable or disable a fax routing method for a specific fax device. The server MUST validate that the client has access to enable or disable routing methods. The *RoutingGUID* parameter MUST be for a valid routing method.

```
error_status_t FAX_EnableRoutingMethod(
  [in] RPC_FAX_PORT_HANDLE FaxPortHandle,
  [in, string, unique] LPCWSTR RoutingGuid,
  [in] BOOL Enabled
);
```

FaxPortHandle: A remote procedure call (RPC) context handle that references a specified fax port. This parameter MUST NOT be NULL.

RoutingGuid: A pointer to a constant, null-terminated character string that MUST specify the globally unique identifier (GUID) that uniquely identifies the fax routing method on which to act.

Enabled: A Boolean variable that indicates whether the application is enabling or disabling the fax routing method that is specified by the *RoutingGuid* parameter. If this parameter is **TRUE**,

the application is requesting that the server enable the routing method; if this parameter is **FALSE**, the application is requesting that the server disable the routing method.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below:

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000000D ERROR_INVALID_DATA	The data is invalid.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.15 FAX_EndCopy (Opnum 72)

The **FAX_EndCopy (Opnum 72)** method is called by the client to end a copy process from or to the server.

On success, the server MUST terminate the specified copy operation previously begun with **FAX_StartCopyToServer (Opnum 68)** or **FAX_StartCopyMessageFromServer (Opnum 69)**.

```
error_status_t FAX_EndCopy(
  [in, out, ref] PRPC_FAX_COPY_HANDLE lphCopy
);
```

lphCopy: The lphCopy handle MUST be the one returned by **FAX_StartCopyToServer (Opnum 68)** or **FAX_StartCopyMessageFromServer (Opnum 69)**.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.16 FAX_EndMessagesEnum (Opnum 64)

The **FAX_EndMessagesEnum (Opnum 64)** method is called by the client.

On success, the server MUST halt the enumerating of messages in the specified archives.

```
error_status_t FAX_EndMessagesEnum(
  [in, out, ref] PRPC_FAX_MSG_ENUM_HANDLE lpHandle
);
```

lpHandle: The parameter lpHandle MUST have been returned by **FAX_StartMessagesEnum (Opnum 63)**.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x000010DD ERROR_INVALID_OPERATION	The operation identifier is not valid.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.17 FAX_EndServerNotification (Opnum 75)

The **FAX_EndServerNotification (Opnum 75)** method is called by the client to stop the notifications from the server, which were initiated by a call to **FAX_StartServerNotification (Opnum 73)**, **FAX_StartServerNotificationEx (Opnum 74)**, or **FAX_StartServerNotificationEx2 (Opnum 92)**.

On success, the server MUST stop notifying the client of events.

```
error_status_t FAX_EndServerNotification(
  [in, out, ref] PRPC_FAX_EVENT_EX_HANDLE lpHandle
);
```

lpHandle: A pointer to a previously registered context handle. The *lpHanlde* parameter MUST match the one supplied to the server when using the **FAX_StartServerNotification** (section 3.1.4.100) family of calls.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below:

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x000010DD ERROR_INVALID_OPERATION	The operation identifier is not valid.

To stop notifications, the client calls **FAX_EndServerNotification (Opnum 75)**; the server should call **FAX_CloseConnection (Opnum 2)** to close the connection.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.18 FAX_EnumAccounts (Opnum 95)

The **FAX_EnumAccounts (Opnum 95)** method is called by the client to enumerate all the fax accounts on the server.

The server MUST validate that the user has the access to enumerate accounts. The *Buffer*, *BufferSize*, and *lpdwAccounts* parameters MUST NOT be NULL. On success, the server MUST enumerate all existing fax accounts and return the enumerated accounts in *Buffer*.<12>

```
error_status_t FAX_EnumAccounts(
  [in] handle_t hBinding,
  [in] DWORD level,
  [out, size_is(,*BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize,
  [out, ref] LPDWORD lpdwAccounts
);
```

hBinding: The fax server handle that is returned by a call to the **Fax_ConnectFaxServer (Opnum 80)** method. This is a context handle.

level: A **DWORD** value that indicates the type of structure that is pointed to by *Buffer*. The value passed in this parameter MUST be zero.

Buffer: A pointer to a **FAX_ACCOUNT_INFO_0 (section 2.2.18)** structure that contains fax account information.

BufferSize: A pointer to a **DWORD** value that specifies the size, in bytes, of the buffer that is pointed to by the *Buffer* parameter.

lpdwAccounts: A DWORD that contains the number of accounts whose information is being returned.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00000103 ERROR_NO_MORE_ITEMS	No more data is available.
0x00001B59 FAX_ERR_SRV_OUTOFMEMORY	The fax server failed to allocate memory.

The account name that *lpcwstrAccountName* indicates MUST be in one of the following formats. Any other format is invalid.

Format	Description
<machine_name>\<user_name>	For a local user with machine_name as the local machine's name.
<domain_name>\<user_name>	For a nonlocal user.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.19 FAX_EnumerateProviders (Opnum 45)

The **FAX_EnumerateProviders (Opnum 45)** method is called by the client to enumerate all the fax service providers (FSP) installed on the server.

The server MUST validate that the user has the access to enumerate providers. The Buffer parameter MUST NOT be NULL. On success, the server MUST return the fax service providers (FSP) installed on the fax server.

The client SHOULD free the returned buffer.

```
error_status_t FAX_EnumerateProviders(
  [in] handle_t hFaxHandle,
  [out, size_is(*BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize,
  [out, ref] LPDWORD lpdwNumProviders
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Buffer: A pointer to the address of a buffer to receive an array of **FAX_DEVICE_PROVIDER_INFO (section 2.2.22)** structures. Each structure contains information about one fax device provider, as it pertains to the entire fax service.

BufferSize: A pointer to a **DWORD** in which to return the size, in bytes, of the buffer.

lpdwNumProviders: A pointer to a **DWORD** variable to receive the number of **FAX_ENUM_PROVIDER_STATUS (section 2.2.44)** structures that the method returns in the *Buffer* parameter. This number MUST be equal to the total number of FSPs installed on the target server.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00001B59 FAX_ERR_SRV_OUTOFMEMORY	The fax server failed to allocate memory.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.20 FAX_EnumGlobalRoutingInfo (Opnum 17)

The **FAX_EnumGlobalRoutingInfo (Opnum 17)** method is called by the client to enumerate global routing information.

The server MUST validate that the user has the access to enumerate the global routing information. On success, the server MUST return all the fax routing methods associated with a specific fax server in RoutingInfoBuffer.

The client SHOULD free the returned buffer.

```
error_status_t FAX_EnumGlobalRoutingInfo(
  [in] handle_t hBinding,
  [out, size_is(, *RoutingInfoBufferSize)]
    LPBYTE* RoutingInfoBuffer,
  [out, ref] LPDWORD RoutingInfoBufferSize,
  [out, ref] LPDWORD MethodsReturned
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

RoutingInfoBuffer: A pointer to the address of a buffer to receive an array of **FAX_GLOBAL_ROUTING_INFOW (section 2.2.24)** structures. Each structure contains information about one fax routing method, as it pertains to the entire Fax Service.

RoutingInfoBufferSize: A variable to return the size, in bytes, of the routing information buffer.

MethodsReturned: A pointer to a **DWORD** variable to receive the number of **FAX_GLOBAL_ROUTING_INFOW** (section 2.2.24) structures that the method returns in the *RoutingInfoBuffer* parameter. This number should equal the total number of fax routing methods installed on the target server.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000001 ERROR_INVALID_FUNCTION	The function is incorrect.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.21 FAX_EnumJobs (Opnum 4)

The **FAX_EnumJobs (Opnum 4)** method is called by the client to enumerate all the fax jobs on the specified fax server.

In response, the server MUST validate if the client has access to enumerate the jobs. On success, the server MUST return information about all the queued and active jobs in *Buffer*. It MUST also return the total size of the buffer in which the information is returned and the total number of enumerated jobs.

The client SHOULD free the returned buffer.

```
error_status_t FAX_EnumJobs(
  [in] handle_t hBinding,
  [out, size_is(,*BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize,
  [out, ref] LPDWORD JobsReturned
```

```
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Buffer: A pointer to the address of a buffer to receive an array of **FAX_JOB_ENTRY (section 2.2.4)** structures.

BufferSize: A variable to return the size, in bytes, of the job information buffer.

JobsReturned: A pointer to a **DWORD** variable to receive the number of **FAX_JOB_ENTRY** (section 2.2.4) structures that the method returns in the *Buffer* parameter.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.22 FAX_EnumJobsEx (Opnum 28)

The **FAX_EnumJobsEx (Opnum 28)** method is called by the client to enumerate a specified set of jobs on the server's queue. The type of jobs to enumerate is described by the dwJobTypes argument.

In response, the server MUST validate if the client has access to enumerate the jobs. On success, the server MUST return information about all the jobs of the specified type. It MUST also return the total size of the buffer in which the information is returned and the total number of enumerated jobs.

The client SHOULD free the returned buffer.

```
error_status_t FAX_EnumJobsEx(
  [in] handle_t hBinding,
  [in] DWORD dwJobTypes,
  [out, size_is(, *BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize,
  [out, ref] LPDWORD lpdwJobs
```

```
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwJobTypes: A **DWORD** value that MUST be a bitwise combination of job types. Only jobs that are of the requested types should be returned in the buffer.

Value	Meaning
JT_UNKNOWN 0x00000001	The job type is unknown. This value indicates that the fax server has not yet scheduled the job.
JT_SEND 0x00000002	The job is an outgoing fax transmission.
JT_RECEIVE 0x00000004	The job is an incoming fax transmission.
JT_ROUTING 0x00000008	The fax server tried to route the fax transmission, but routing failed. The fax server will attempt to route the job again.
JT_FAIL_RECEIVE 0x00000010	The fax server failed to receive the job. This value is included for backward compatibility.
JT_BROADCAST 0x00000020	The job is an outgoing broadcast message.

Buffer: A pointer to the address of a buffer to receive an array of **FAX_JOB_ENTRY_EXW (section 2.2.26)** structures. Each structure describes one fax job. This data is serialized on the wire. The field length must be clamped to 32 bits before serialization. If the data is a pointer to a structure, the number of bytes equal to the size of the structure are allocated and are replaced by the corresponding pointer.

BufferSize: A variable to return the size, in bytes, of the buffer.

lpdwJobs: A pointer to a **DWORD** variable to receive the number of **FAX_JOB_ENTRY_EXW** (section 2.2.26) structures that the method returns in the *Buffer* parameter.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057	The parameter is incorrect.

Return value/code	Description
ERROR_INVALID_PARAMETER	

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.23 FAX_EnumJobsEx2 (Opnum 88)

The **FAX_EnumJobsEx2 (Opnum 88)** method is called by the client to enumerate a specified set of jobs on the server's queue for a specific fax account. The type of jobs to enumerate is described by the dwJobTypes argument.

In response, the server MUST validate if the client has access to enumerate the jobs for the specified account. If the lpcwstrAccountName is not NULL, then the server MUST validate the format of the account name. It MUST also verify that the level argument is set to 1.

On success, the server MUST return information about all the jobs of the specified type for the specified account. It MUST also return the total size of the buffer in which the information is returned and the total number of jobs enumerated.

The client SHOULD free the returned buffer.<13>

```
error_status_t FAX_EnumJobsEx2(
  [in] handle_t hBinding,
  [in] BOOL fAllAccounts,
  [in, string, unique] LPCWSTR lpcwstrAccountName,
  [in] DWORD dwJobTypes,
  [in] DWORD level,
  [out, size_is(,*BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize,
  [out, ref] LPDWORD lpdwJobs
);
```

hBinding: Fax server handle that is returned by a call to the **Fax_ConnectFaxServer (Opnum 80) (section 3.1.4.10)** method. This is a context handle.

fAllAccounts: Flag indicating whether the jobs for all accounts are enumerated. If this parameter is non-zero, the jobs for all accounts are enumerated; otherwise, *lpcwstrAccountName* SHOULD indicate which accounts are to be enumerated.

lpcwstrAccountName: Pointer to a constant, null-terminated character string that indicates which account to enumerate. If this value is set to **NULL**, the current account's jobs are enumerated. Cross-account enumeration is currently not supported.

dwJobTypes: A **DWORD** value that MUST consist of a bitwise combination of job types. Only jobs that are of the requested types should be returned in the buffer.

Value	Meaning
JT_UNKNOWN 0x00000001	Job type is unknown. This value indicates that the fax server has not yet scheduled the job.
JT_SEND	Job is an outgoing fax transmission.

Value	Meaning
0x00000002	
JT_RECEIVE 0x00000004	Job is an incoming fax transmission.
JT_ROUTING 0x00000008	Fax server tried to route the fax transmission, but routing failed. The fax server will attempt to route the job again.
JT_FAIL_RECEIVE 0x00000010	Fax server failed to receive the job. This value is included for backward compatibility.
JT_BROADCAST 0x00000020	Job is an outgoing broadcast message.

level: A **DWORD** value that indicates the type of structure to return in *Buffer*. The value MUST be set to 1.

Buffer: Pointer to the address of a buffer that will receive an array of **FAX_JOB_ENTRY_EX_1 (section 2.2.25)** structures. Each structure describes one fax job.

BufferSize: Pointer to a **DWORD** value that returns the size of *Buffer* in bytes.

lpdwJobs: Pointer to a **DWORD** value that receives the number of **FAX_JOB_ENTRY_EX_1** (section 2.2.25) structures that the method returns in the *Buffer* parameter.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the Fax-Specific Errors as defined in section 2.2.39 or one of the standard Windows errors as defined in section 4 of [MS-ERREF], the most common of which are listed below:

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

The account name that *lpcwstrAccountName* indicates MUST be in one of the following formats. Any other format is invalid.

Format	Description
<machine_name>\<user_name>	For a local user with machine_name as the local machine's name.
<domain_name>\<user_name>	For a nonlocal user.

Exceptions Thrown:

No exceptions are thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

3.1.4.24 FAX_EnumMessages (Opnum 65)

The **FAX_EnumMessages (Opnum 65)** method is called by the client.

In response, the server MUST validate that the hEnum argument passed by the client was created as part of a prior **FAX_StartMessagesEnum (Opnum 63) (section 3.1.4.98)** call. The server MUST validate that the dwNumMessages argument is not zero.

On success, the server MUST return information about the messages. The server MUST also return the size of the information returned and the number of messages for which it could successfully retrieve the information. The latter value MUST NOT exceed dwNumMessages.

The client SHOULD free the returned buffer.

```
error_status_t FAX_EnumMessages(
  [in, ref] RPC_FAX_MSG_ENUM_HANDLE hEnum,
  [in] DWORD dwNumMessages,
  [out, size_is(, *lpdwBufferSize)]
    LPBYTE* lppBuffer,
  [out, ref] LPDWORD lpdwBufferSize,
  [out, ref] LPDWORD lpdwNumMessagesRetrieved
);
```

hEnum: The handle that is provided by the client RPC layer when the RPC call is made. The client sends the handle previously returned to the client by a call to **FAX_StartMessagesEnum (Opnum 63)** (section 3.1.4.98).

dwNumMessages: A **DWORD** value indicating the maximum number of messages the caller requires to enumerate. This value MUST NOT be zero.

lppBuffer: A pointer to a buffer of **FAX_MESSAGE (section 2.2.29)** structures. This buffer contains *lpdwReturnedMsgs* entries.

lpdwBufferSize: A pointer to a **DWORD** in which to return the size, in bytes, of the buffer.

lpdwNumMessagesRetrieved: A pointer to a **DWORD** value indicating the actual number of messages retrieved. This value should not exceed *dwNumMessages*.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below:

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Return value/code	Description
0x00000103 ERROR_NO_MORE_ITEMS	No more data is available.
0x00001B59 FAX_ERR_SRV_OUTOFMEMORY	The fax server failed to allocate memory.

The client expects that this method is incremental and uses an internal context cursor to point to the next set of messages to retrieve for each call. The cursor is set to point to the beginning of the messages in the archive after a successful call to **FAX_StartMessagesEnum** (section 3.1.4.98). Each successful call to **FAX_EnumMessages** (section 3.1.4.24) advances the cursor by the number of messages retrieved. After the cursor reaches the end of the enumeration, the method fails with the 232 (ERROR_NO_DATA) error code. The **FAX_EndMessagesEnum (section 3.1.4.16)** method should then be called.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.25 FAX_EnumMessagesEx (Opnum 91)

The **FAX_EnumMessagesEx (Opnum 91)** method is called by the client. This message differs from the **FAX_EnumMessages (section 3.1.4.24)** in that this function takes a level parameter, which differentiates the type of message information structure that the function returns.

In response, the server MUST validate that the hEnum argument that is passed by the client was created as part of a prior **FAX_StartMessagesEnum (Opnum 63)** call. The server MUST validate that the dwNumMessages argument is not zero.

On success, the server MUST return information regarding messages. The server MUST return the size of the returned information and the number of messages for which it could successfully retrieve the information. The latter value MUST NOT exceed dwNumMessages. The server MUST return the level of information. This return value is decided by whether the client used **FAX_StartMessagesEnum (Opnum 63)** or **FAX_StartMessagesEnumEx (Opnum 90)** to start the enumeration of messages.

The client SHOULD free the returned buffer.<14>

```
error_status_t FAX_EnumMessagesEx(
  [in, ref] RPC_FAX_MSG_ENUM_HANDLE hEnum,
  [in] DWORD dwNumMessages,
  [out, size_is(,*lpdwBufferSize)]
    LPBYTE* lppBuffer,
  [out, ref] LPDWORD lpdwBufferSize,
  [out, ref] LPDWORD lpdwNumMessagesRetrieved,
  [out, ref] LPDWORD lpdwLevel
);
```

hEnum: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwNumMessages: A **DWORD** value that indicates the maximum number of messages that the caller requires to enumerate. This value MUST NOT be zero.

lppBuffer: A pointer to a buffer of **FAX_MESSAGE_1 (section 2.2.28)** structures that contain *lpdwReturnedMsgs* entries.

lpdwBufferSize: A pointer to a **DWORD** value that specifies the size, in bytes, of the buffer.

lpdwNumMessagesRetrieved: A pointer to a **DWORD** value that indicates the actual number of retrieved messages. This value should not exceed *dwNumMessages*.

lpdwLevel: A pointer to a **DWORD** value that indicates the structure to return.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00000103 ERROR_NO_MORE_ITEMS	No more data is available.
0x00001B59 FAX_ERR_SRV_OUTOFMEMORY	The fax server failed to allocate memory.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.26 FAX_EnumOutboundGroups (Opnum 54)

The **FAX_EnumOutboundGroups (Opnum 54)** method is called by the client.

In response, the server MUST validate that the client has access to enumerate the outbound routing groups.

On success, the server MUST return information about all its outbound routing groups in ppData. It MUST also return the size of the information returned and the number of outbound routing groups for which it enumerated information successfully.

The client SHOULD free lppData buffer.

```
error_status_t FAX_EnumOutboundGroups(
  [in] handle_t hFaxHandle,
  [out, size_is(, *lpdwDataSize)]
    LPBYTE* ppData,
  [out, ref] LPDWORD lpdwDataSize,
  [out, ref] LPDWORD lpdwNumGroups
);
```

hFaxHandle: A handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

ppData: The address of a pointer to a buffer of **RPC_FAX_OUTBOUND_ROUTING_GROUPW (section 2.2.30)** structures.

lpdwDataSize: The size, in bytes, of the returned *ppData* buffer.

lpdwNumGroups: The number of groups that are returned.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below:

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00001B63 FAX_ERR_NOT_SUPPORTED_ON_THIS_SKU	The fax server API version does not support the requested operation.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.27 FAX_EnumOutboundRules (Opnum 59)

The **FAX_EnumOutboundRules (Opnum 59)** method is called by the client to enumerate all the outbound routing rules that are present on the specified fax server.

In response, the server MUST validate if the client has access to enumerate the outbound routing rules.

On success, the server MUST return information about all its outbound routing rules in lppData. It MUST also return the size of the returned information and the number of outbound routing rules for which it successfully enumerated information.

The client SHOULD free lppData.

```
error_status_t FAX_EnumOutboundRules(
  [in] handle_t hFaxHandle,
  [out, size_is(, *lpdwDataSize)]
    LPBYTE* ppData,
  [out, ref] LPDWORD lpdwDataSize,
  [out, ref] LPDWORD lpdwNumRules
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

ppData: A pointer to a buffer of **RPC_FAX_OUTBOUND_ROUTING_RULEW (section 2.2.31)** structures.

lpdwDataSize: A pointer to a **DWORD** in which to return the size, in bytes, of the *ppData* buffer.

lpdwNumRules: A pointer to a **DWORD** value indicating the number of rules retrieved.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00001B63 FAX_ERR_NOT_SUPPORTED_ON_THIS_SKU	The fax server API version does not support the requested operation.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.28 FAX_EnumPorts (Opnum 10)

The **FAX_EnumPorts (Opnum 10)** method is called by the client to obtain port state information.

In response, the server MUST validate if the client has access to enumerate all the devices (ports) on the server. On success, the server MUST return information about all its devices in *PortBuffer*. It MUST also return the size of the returned information and the number of devices for which it successfully enumerated information.

The client SHOULD free the returned buffer.

```
error_status_t FAX_EnumPorts(
  [in] handle_t hBinding,
  [out, size_is(,*BufferSize)] LPBYTE* PortBuffer,
  [out, ref] LPDWORD BufferSize,
  [out, ref] LPDWORD PortsReturned
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

PortBuffer: A pointer to the address of a buffer to receive an array of **FAX_PORT_INFO (section 2.2.5)** structures. Each structure describes one fax port.

BufferSize: A variable to return the size, in bytes, of the port buffer.

PortsReturned: A pointer to a **DWORD** variable to receive the number of **FAX_PORT_INFO** (section 2.2.5) structures that the method returns in the *PortBuffer* parameter.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.29 FAX_EnumPortsEx (Opnum 48)

The **FAX_EnumPortsEx (Opnum 48)** method is called by the client to enumerate detailed port state information for each device that is connected to the fax server.

In response, the server MUST validate if the client has access to enumerate all the devices (ports) on the server. On success, the server MUST return information about all its devices in Buffer. It MUST also return the size of the returned information and the number of devices for which it successfully enumerated information.

The client SHOULD free the returned buffer.

```
error_status_t FAX_EnumPortsEx(
  [in] handle_t hFaxHandle,
  [out, size_is(, *BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize,
  [out, ref] LPDWORD lpdwNumPorts
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Buffer: A pointer to the address of a buffer to receive an array of **FAX_PORT_INFO_EXW (section 2.2.34)** structures. Each structure describes one fax port. The data includes, among other items, the permanent line identifier, and the current status and capability of the port.

BufferSize: A pointer to a **DWORD** in which to return the size, in bytes, of the buffer.

lpdwNumPorts: A pointer to a **DWORD** variable that receives the number of ports that are returned by the method.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.30 FAX_EnumRoutingExtensions (Opnum 78)

The **FAX_EnumRoutingExtensions (Opnum 78)** function is called by the client to enumerate all the routing extensions that are registered with the specified fax server. The function returns detailed information about each of the routing extensions.

In response, the server MUST validate if the client has access to enumerate all the routing extensions on the server. On success, the server MUST return information about all its routing extensions in *Buffer*. It MUST also return the size of the returned information and the number of routing extensions for which it successfully enumerated information.

The client SHOULD free the returned buffer.

```
error_status_t FAX_EnumRoutingExtensions(
  [in] handle_t hFaxHandle,
  [out, size_is(, *BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize,
  [out, ref] LPDWORD lpdwNumExts
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Buffer: A pointer to the address of a buffer to receive an array of **FAX_ROUTING_EXTENSION_INFO (section 2.2.36)** structures. Each structure contains information about one fax routing extension, as it pertains to the entire Fax Service.

BufferSize: A pointer to a **DWORD** in which to return the size, in bytes, of the buffer.

lpdwNumExts: A pointer to a **DWORD** variable to receive the number of **FAX_ROUTING_EXTENSION_INFO** (section 2.2.36) structures that the method returns in the *ppRoutingExtensions* parameter. This number MUST equal the total number of fax device routing extensions that are installed on the target server.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.31 FAX_EnumRoutingMethods (Opnum 13)

The **FAX_EnumRoutingMethods (Opnum 13)** method is called by the client to enumerate all the routing methods that are registered with the specified fax server. The client calls **FaxOpenPort (Opnum 2) (section 3.1.4.65)** to get the value for FaxPortHandle. The function returns detailed information about each of the routing methods.

In response, the server MUST validate that the client has access to query configuration. It MUST allocate memory for the routing information array to be passed out and fill it with data.

On success, the server MUST fill the buffer with the routing information for the particular port, along with the buffer size and the number of enumerated methods.

The client SHOULD free the buffer by calling FaxFreeBuffer.<15>

```
error_status_t FAX_EnumRoutingMethods(
  [in] RPC_FAX_PORT_HANDLE FaxPortHandle,
  [out, size_is(, *RoutingInfoBufferSize)]
    LPBYTE* RoutingInfoBuffer,
  [out, ref] LPDWORD RoutingInfoBufferSize,
  [out, ref] LPDWORD PortsReturned
);
```

FaxPortHandle: A remote procedure call (RPC) context handle that references a specified fax port.

RoutingInfoBuffer: A pointer to the address of a buffer to receive an array of **FAX_ROUTING_METHOD (section 2.2.6)** structures. Each structure contains information about one fax routing method.

RoutingInfoBufferSize: A variable to return the size, in bytes, of the routing method buffer.

PortsReturned: A pointer to a **DWORD** variable to receive the number of **FAX_ROUTING_METHOD** (section 2.2.6) structures that are returned by the *RoutingInfoBuffer* parameter.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below:

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000001 ERROR_INVALID_FUNCTION	The function is incorrect.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x0000000D ERROR_INVALID_DATA	The data is invalid.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.32 FAX_GetAccountInfo (Opnum 96)

The **FAX_GetAccountInfo (Opnum 96)** method is called by the client to retrieve information about a specified account.

In response, the server MUST validate the account name that is passed in. If the account name that is passed in is not the same as the logged-on user, the server MUST check whether the client has access to query configuration. The server MUST then allocate the buffer to hold the account information.

On success, the server MUST return the detailed information about the account that is passed in the buffer as per the level specified, along with the buffer size.

The client SHOULD free the buffer by calling FaxFreeBuffer.<16>

```
error_status_t FAX_GetAccountInfo(
  [in] handle_t hBinding,
  [in, string, unique] LPCWSTR lpcwstrAccountName,
```

```
  [in] DWORD level,
  [out, size_is(,*BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize
);
```

hBinding: The fax server handle that is returned by a call to the **Fax_ConnectFaxServer (Opnum 96)** method. This is a context handle.

lpcwstrAccountName: A pointer to a constant, null-terminated character string that contains the name of the account for which to retrieve information.

level: A **DWORD** value that indicates the type of structure that is pointed to by *Buffer*. This MUST be zero.

Buffer: A pointer to a **FAX_ACCOUNT_INFO_0 (section 2.2.18)** structure that contains fax account information.

BufferSize: A pointer to a **DWORD** value that specifies the size, in bytes, of the structure that is pointed to by the *Buffer* parameter.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00000649 ERROR_INVALID_HANDLE	The handle is in an invalid state.

The account name that *lpcwstrAccountName* indicates must be in one of the following formats. Any other format is invalid.

Format	Description
<machine_name>\<user_name>	For a local user that has machine_name as the local machine's name.
<domain_name>\<user_name>	For a nonlocal user.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.33 FAX_GetActivityLoggingConfiguration (Opnum 43)

The **FAX_GetActivityLoggingConfiguration (Opnum 43)** method is called by the client to retrieve the current activity logging configuration.

In response, the server MUST validate that the client has access to query configuration. It MUST then allocate memory for the activity logging information to be passed out and fill it with data.

To indicate success, the server MUST return the buffer that contains the activity logging information, along with the buffer size.

The client SHOULD free the buffer.

```
error_status_t FAX_GetActivityLoggingConfiguration(
  [in] handle_t hFaxHandle,
  [out, size_is(, *BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Buffer: A pointer to a **FAX_ACTIVITY_LOGGING_CONFIGW (section 2.2.19)** structure.

BufferSize: A pointer to a **DWORD** in which to return the size, in bytes, of the buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) to indicate success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.34 FAX_GetArchiveConfiguration (Opnum 41)

The **FAX_GetArchiveConfiguration (Opnum 41)** method is called by the client to retrieve the current archive configuration on the fax server. In response, the server returns archive configuration information of the fax server.

In response, the server MUST validate that the client has access to query configuration. It MUST then allocate memory for the archive configuration information to be passed out and fill it with data.

To indicate success, the server MUST return the buffer that contains the archive configuration information, along with the buffer size.

The client SHOULD free the buffer by calling FaxFreeBuffer.

```
error_status_t FAX_GetArchiveConfiguration(
  [in] handle_t hFaxHandle,
  [in] FAX_ENUM_MESSAGE_FOLDER Folder,
  [out, size_is(, *BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize
);
```

hFaxHandle: A remote procedure call (RPC) context handle that references a specified fax port.

Folder: Archive location. This MUST be either **FAX_ENUM_MESSAGE_FOLDER (section 2.2.1)** or **FAX_MESSAGE_FOLDER_SENTITEMS** (section 2.2.1).

Buffer: A pointer to a **FAX_ARCHIVE_CONFIGW (section 2.2.20)** object.

BufferSize: A pointer to a **DWORD** in which to return the size, in bytes, of the buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) to indicate success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x000010DD ERROR_INVALID_OPERATION	The operation identifier is not valid.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.35 FAX_GetConfigOption (Opnum 104)

The **FAX_GetConfigOption (Opnum 104)** is called by the client to retrieve a configuration setting at the server using an RPC_REQUEST packet.

In response, the server MUST validate that the client has access as follows. Use of this method does NOT require FAX_ACCESS_QUERY_CONFIG access rights. A calling user with any access-control entry (ACE) on the server can use this method.

On success, the appropriate config option MUST be passed out by the server.<17>

```
error_status_t FAX_GetConfigOption(
  [in] handle_t hFaxHandle,
  [in] FAX_ENUM_CONFIG_OPTION option,
  [out] LPDWORD lpdwValue
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is initiated.

option: Identifies the configuration option to be returned. This parameter MUST be a value from the **FAX_ENUM_CONFIG_OPTION (section 2.2.2)** enumeration.

lpdwValue: A pointer to a **DWORD** that holds the value of the configuration option upon return. The value's type depends on the configuration option that was asked for using the *option* parameter.

If *option* was set to FAX_CONFIG_OPTION_ALLOW_PERSONAL_CP, *lpdwValue* contains a **BOOL** that MUST take one of the following values.

Value	Meaning
TRUE 0x00000001	The server allows personal cover pages.
FALSE 0x00000000	The server allows only server-side cover pages.

If *option* was set to FAX_CONFIG_OPTION_QUEUE_STATE, *lpdwValue* is a **DWORD** value that MUST specify state information about the fax server queue. If this value is zero, both the incoming and outgoing queues are unblocked. Otherwise, this value MUST be a combination of one or more of the following flags.

Value	Meaning
0x00000000	Both the incoming and outgoing queues are unblocked.
FAX_INCOMING_BLOCKED 0x00000001	The incoming faxes queue is blocked. The fax server does not answer any new incoming faxes.
FAX_OUTBOX_BLOCKED 0x00000002	The outbox queue is blocked. The fax server does not accept submission of new faxes. If the outbox is not paused, faxes in the queue are being processed.
FAX_OUTBOX_PAUSED 0x00000004	The outbox queue is paused. The fax server will not start sending outgoing faxes from the queue. Fax transmissions in progress are not affected. If the outbox is not blocked, the fax server still accepts submission of new faxes to the queue.

If *option* was set to FAX_CONFIG_OPTION_ALLOWED_RECEIPTS, *lpdwValue* contains a **DWORD** that MUST be a bitwise combination of one or more of the flags that are specified in **FAX_ENUM_DELIVERY_REPORT_TYPES (section 2.2.62)**.

If *option* was set to FAX_CONFIG_OPTION_INCOMING_FAXES_PUBLIC, *lpdwValue* contains a **BOOL** that MUST take one of the following values.

Value	Meaning
TRUE 0x00000001	All incoming faxes can be viewed by all fax users.
FALSE 0x00000000	Incoming faxes can be viewed only by recipients.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The *option* parameter is an invalid value or the *lpdwValue* parameter is **NULL**.

Use of this method does not require FAX_ACCESS_QUERY_CONFIG access rights. A calling user with any access-control entry (ACE) on the server can use this method.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.36 FAX_GetConfiguration (Opnum 19)

The **FAX_GetConfiguration (Opnum 19)** method is called by the client.

In response, the server MUST validate that the client has access to query configuration. It MUST then allocate memory for the configuration information to be passed out and fill it with data.

To indicate success, the server MUST return the buffer that contains the configuration information, along with the buffer size.

The client SHOULD free the buffer.

```
error_status_t FAX_GetConfiguration(
  [in] handle_t hBinding,
  [out, size_is(, *BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Buffer: A pointer to the address of a buffer to receive a **FAX_CONFIGURATION (section 2.2.21)** structure. The structure contains the current configuration settings for the fax server.

BufferSize: A variable to return the size, in bytes, of the buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

If the size of the buffer is not big enough, the method should return an error.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.37 FAX_GetCountryList (Opnum 30)

The **FAX_GetCountryList (Opnum 30)** method is called by the client to retrieve the list of country/region information that is defined on the server. TAPI maintains this list, which contains information like the country/region name or country/region ID.

In response, the server MUST validate that the client has any access to the fax server. It MUST then allocate memory for the country/region list array to be passed out and fill it with data. To indicate success, the server MUST return the buffer that contains the country/region list, along with the buffer size.

The client SHOULD free the buffer.

```
error_status_t FAX_GetCountryList(
  [in] handle_t FaxHandle,
  [out, size_is(, *BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize
);
```

FaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Buffer: A pointer to a buffer of type **FAX_TAPI_LINECOUNTRY_LIST (section 2.2.38)** in which to place the country/region information.

BufferSize: A pointer to a **DWORD** in which to return the size, in bytes, of the buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.38 FAX_GetDeviceStatus (Opnum 8)

The **FAX_GetDeviceStatus (Opnum 8)** method is called by the client to retrieve information about a specified fax device (port).

In response, the server MUST validate that the client has access to query configuration. The server MUST validate that the FaxPortHandle is not NULL. It MUST then allocate memory for the status buffer to be passed out and fill it with data.

To indicate success, the server MUST return the buffer that contains the status information, along with the buffer size.

The client SHOULD free the buffer.

```
error_status_t FAX_GetDeviceStatus(
  [in] RPC_FAX_PORT_HANDLE FaxPortHandle,
  [out, size_is(,*BufferSize)] LPBYTE* StatusBuffer,
  [out, ref] LPDWORD BufferSize
);
```

FaxPortHandle: A remote procedure call (RPC) context handle that references a specified fax port.

StatusBuffer: A pointer to the address of a buffer to receive a **FAX_DEVICE_STATUS (section 2.2.7)** structure. The structure describes the status of one fax device.

BufferSize: A variable to return the size, in bytes, of the job information buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x0000000D ERROR_INVALID_DATA	The data is invalid.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.39 FAX_GetExtensionData (Opnum 49)

The **FAX_GetExtensionData (Opnum 49)** method is called by the client to retrieve a fax extension's private data. The lpctstrNameGUID MUST be for a valid routing extension for which the client requests the private data.

In response, the server MUST validate that the client has access to query configuration. It MUST then allocate memory for the extension data to be passed out and fill it with data.

To indicate success, the server MUST return the buffer that contains the extension data, along with the buffer size.

The client SHOULD free the buffer.

```
error_status_t FAX_GetExtensionData(
  [in] handle_t hFaxHandle,
  [in] DWORD dwDeviceId,
  [in, string, ref] LPCWSTR lpctstrNameGUID,
  [out, size_is(, *lpdwDataSize)]
    LPBYTE* ppData,
  [out, ref] LPDWORD lpdwDataSize
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwDeviceId: The device identifier. A value of zero indicates the caller requests a named data BLOB that is not associated with any specific device. This value can be used to store configurations that affect all the devices. For example, an Optical Character Recognition (OCR) routing extension might export several different routing methods, which rely on the same OCR parameters. This routing extension should associate the OCR configuration with a non-specific device so that it would become global.

lpctstrNameGUID: A pointer to a null-terminated string that contains a globally unique identifier (GUID) that identifies the data to return; for example, {b8959fc9-4e77-4ee9-8411-009acb1bbf3e}. This GUID corresponds to a named value.

ppData: A pointer to an allocated private data buffer. This buffer contains the data that is returned by the Fax Service Provider.

lpdwDataSize: A pointer to a **DWORD** value that returns the size, in bytes, of the data that is pointed to by the *ppData* parameter.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.40 FAX_GetGeneralConfiguration (Opnum 97)

The **FAX_GetGeneralConfiguration (Opnum 97)** method is called by the client to request information about the general configuration at the server.

In response, the server MUST validate that the client has access to query configuration. It then MUST allocate memory for the configuration information to be passed out and then fill it with data.

On success, the server MUST return the buffer that contains the configuration information as specified by the level, along with the buffer size.

The client SHOULD free the buffer.<18>

```
error_status_t FAX_GetGeneralConfiguration(
  [in] handle_t hBinding,
  [in] DWORD level,
  [out, size_is(,*BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize
);
```

hBinding: The fax server handle that is returned by a call to the **FAX_ConnectFaxServer (section 3.1.4.10)** method. This is a context handle.

level: A **DWORD** value that indicates the type of structure pointed to by *Buffer*. This MUST be zero.

Buffer: A pointer to a **FAX_GENERAL_CONFIG (section 2.2.23)** structure that contains the server information to retrieve.

BufferSize: A pointer to a **DWORD** value that specifies the size, in bytes, of the buffer that is pointed to by the *Buffer* parameter.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.41 FAX_GetJob (Opnum 5)

The **FAX_GetJob (Opnum 5)** method is called by the client to retrieve information regarding a specific job. The job is specified by the *JobId*.

In response the server MUST validate that the *JobId* is for a valid job. The server MUST validate that the client has read access to the job.

On success, the server MUST return the job information of the specified queued or active job along with the size.

The client SHOULD free the returned buffer.

```
error_status_t FAX_GetJob(
  [in] handle_t hBinding,
  [in] DWORD JobId,
  [out, size_is(,*BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

JobId: A unique number that identifies a queued or active fax job.

Buffer: A pointer to the address of a buffer to receive an array of **FAX_JOB_ENTRY (section 2.2.4)** structures.

BufferSize: A variable to return the size, in bytes, of the job information buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00001B61 FAX_ERR_MESSAGE_NOT_FOUND	The fax server cannot complete the operation because the number of active fax devices allowed for this version of Windows was exceeded.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.42 FAX_GetJobEx (Opnum 29)

The **FAX_GetJobEx (Opnum 29)** is called by the client to retrieve information about a specified job at the server. The job is identified by the job message ID.

In response the server MUST validate that the message ID is for a valid job. The server MUST validate that the client has read access to the job.

On success, the server MUST return the job information of the queued job and also the size.

This method is an extended version of **FAX_GetJob (Opnum 29)**, which returns a **FAX_JOB_ENTRY_EX (section 2.2.26)** structure for the specified message.

The client SHOULD free the returned buffer.

```
error_status_t FAX_GetJobEx(
  [in] handle_t hBinding,
  [in] DWORDLONG dwlMessageID,
  [out, size_is(, *BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

A unique number that identifies a queued or active fax job. The job MUST be an inbound or outbound transmission.

Buffer: A pointer to the address of a buffer to receive a **FAX_JOB_ENTRY_EX** (section 2.2.26) structure. This data is serialized on the wire. The field length MUST be clamped to 32 bits before serialization. If the data is a pointer to a structure, the number of bytes equal to the size of the structure are allocated and are replaced by the corresponding pointer.

BufferSize: A variable to return the size, in bytes, of the buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00001B61 FAX_ERR_MESSAGE_NOT_FOUND	The fax server cannot complete the operation because the number of active fax devices that are allowed for this version of Windows was exceeded.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.43 FAX_GetJobEx2 (Opnum 87)

The **FAX_GetJobEx2 (Opnum 87)** method is called by the client to retrieve information about a specified job. The job is identified by the job message ID.

In response, the server MUST validate that the message ID is for a valid job. The server MUST validate that the client has read access to the job.

This method is an extended version of **FAX_GetJob (section 3.1.4.41)**, which returns the **FAX_JOB_ENTRY_EX (section 2.2.26)** structure for the specified message.

The client SHOULD free the returned buffer.

```
error_status_t FAX_GetJobEx2(
  [in] handle_t hBinding,
  [in] DWORDLONG dwlMessageID,
  [in] DWORD level,
  [out, size_is(,*BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwlMessageID: A **DWORDLONG** value that specifies a unique number that identifies a queued or active fax job. The job MUST be an inbound or outbound transmission.

level: A **DWORD** value that indicates the structure to return in *Buffer*. This value MUST be set to 1.

Buffer: A pointer to the address of a buffer that receives a **FAX_JOB_ENTRY_EX_1 (section 2.2.25)** structure.

BufferSize: A pointer to a **DWORD** value that specifies the size, in bytes, of the buffer that is pointed to by the *lppBuffer* parameter.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00001B61 FAX_ERR_MESSAGE_NOT_FOUND	The fax server cannot complete the operation because the number of active fax devices that are allowed for this version of Windows was exceeded.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.44 FAX_GetLoggingCategories (Opnum 21)

The **FAX_GetLoggingCategories (Opnum 21)** method is called by the client. In response, the server MUST return the current logging categories for the fax server to which the client has connected. A logging category determines the errors or other events that the fax server records in the application event log.

The client SHOULD free the returned buffer.

```
error_status_t FAX_GetLoggingCategories(
  [in] handle_t hBinding,
  [out, size_is(, *BufferSize)] LPBYTE* Buffer,
  [out, ref] DWORD BufferSize,
  [out, ref] DWORD NumberCategories
```

```
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Buffer: A pointer to the address of a buffer to receive an array of **FAX_LOG_CATEGORY (section 2.2.8)** structures. Each structure describes one current logging category.

BufferSize: A variable to return the size, in bytes, of the buffer.

NumberCategories: A pointer to a **DWORD** variable to receive the number of **FAX_LOG_CATEGORY** structures that the method returns in the *Buffer* parameter.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.45 FAX_GetMessage (Opnum 66)

The **FAX_GetMessage (Opnum 66)** method is called by the client. The archive can be one of the enumerations that are defined by FAX_ENUM_MESSAGE_FOLDER except FAX_MESSAGE_FOLDER_QUEUE.

In response, the server MUST validate that message ID is for a valid message. The server MUST validate that the client has access to read the message.

On success, the server MUST return the contents of the message and also its size.

The client SHOULD free the returned buffer.

```
error_status_t FAX_GetMessage(
  [in] handle_t hFaxHandle,
  [in] DWORDLONG dwlMessageId,
  [in] FAX_ENUM_MESSAGE_FOLDER Folder,
  [out, size_is(, *lpdwBufferSize)]
    LPBYTE* lppBuffer,
```

```
  [out, ref] LPDWORD lpdwBufferSize
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwlMessageId: A **DWORDLONG** value that identifies the fax message to retrieve from the archive.

Folder: The type of archive where the message resides. FAX_MESSAGE_FOLDER_QUEUE is an invalid value for this parameter.

lppBuffer: A pointer to a buffer of **FAX_MESSAGE (section 2.2.29)** structures. This buffer contains the retrieved messages.

lpdwBufferSize: A pointer to a **DWORD** in which to return the size, in bytes, of the buffer that is pointed to by the *lppBuffer* parameter.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00001B61 FAX_ERR_MESSAGE_NOT_FOUND	The fax server cannot complete the operation because the number of active fax devices that are allowed for this version of Windows was exceeded.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.46 FAX_GetMessageEx (Opnum 89)

The **FAX_GetMessageEx (Opnum 89)** method is called by the client to retrieve a particular message from one of the specified fax message archives. The folder value MUST be one of the enumerations that are defined by **FAX_ENUM_MESSAGE_FOLDER (section 2.2.1)** except FAX_MESSAGE_FOLDER_QUEUE. This is an extended version of **FAX_GetMessage (section 3.1.4.45)**, because it takes an additional level parameter supporting the extended structure **FAX_MESSAGE_1 (section 2.2.28)**, as defined in the section that is returned by this function.<19>

In response, the server MUST validate that message ID is for a valid message. The server MUST validate that the client has access to read the message. On success, the server MUST return the contents of the message in *lppBuffer* and also its size.

The client SHOULD free the returned buffer.

```
error_status_t FAX_GetMessageEx(
  [in] handle_t hFaxHandle,
  [in] DWORDLONG dwlMessageId,
  [in] FAX_ENUM_MESSAGE_FOLDER Folder,
  [in] DWORD level,
  [out, size_is(,*lpdwBufferSize)]
    LPBYTE* lppBuffer,
  [out, ref] LPDWORD lpdwBufferSize
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwlMessageId: A **DWORDLONG** value that identifies the fax message to retrieve from the archive.

Folder: A **FAX_ENUM_MESSAGE_FOLDER** (section 2.2.1) enumeration that indicates the type of the archive where the message resides. The FAX_MESSAGE_FOLDER_QUEUE value is invalid for this parameter.

level: A **DWORD** value that indicates the type of structure to return in *lppBuffer*. The only value currently supported is 1.

lppBuffer: A pointer to a buffer of **FAX_MESSAGE_1** (section 2.2.28) structures that contain the retrieved messages.

lpdwBufferSize: A pointer to a **DWORD** value that specifies the size, in bytes, of the buffer that is pointed to by the *lppBuffer* parameter.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00001B61 FAX_ERR_MESSAGE_NOT_FOUND	The fax server cannot complete the operation because the number of active fax devices that are allowed for this version

Return value/code	Description
	of Windows was exceeded.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.47 FAX_GetOutboxConfiguration (Opnum 38)

The **FAX_GetOutboxConfiguration (Opnum 38)** method is called by the client to retrieve the outbox configuration at the server.

In response, the server MUST validate that the client has access to query the outbox configuration. On success, the server MUST return the outbox configuration in Buffer and also its size.

The client SHOULD free the returned buffer.

```
error_status_t FAX_GetOutboxConfiguration(
  [in] handle_t hFaxHandle,
  [out, size_is(, *BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Buffer: A pointer to a **FAX_OUTBOX_CONFIG (section 2.2.12)** object.

BufferSize: A pointer to a **DWORD** in which to return the size, in bytes, of the buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.48 FAX_GetPageData (Opnum 7)

The **FAX_GetPageData (Opnum 7)** method is called by the client to retrieve data in the first page of an outgoing fax job. The information that is returned in the buffer is an in-memory copy of the first page of the TIFF file.

In response, the server MUST validate that the *JobId* is for a valid job. The server MUST validate that the client has read access to the job. On success, the server MUST return the first page of data for the queued or active job in the TIFF 6.0 Class F format in *Buffer*, along with the image width and height.

The client SHOULD free the returned buffer.

For information about TIFF, see [RFC3302].

```
error_status_t FAX_GetPageData(
  [in] handle_t hBinding,
  [in] DWORD JobId,
  [out, size_is(,*BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize,
  [in, out] LPDWORD ImageWidth,
  [in, out] LPDWORD ImageHeight
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

JobId: A unique number that identifies the fax job that is associated with the page of data.

Buffer: A pointer to the address of a buffer to receive the first page of data in the fax document.

BufferSize: A pointer to a **DWORD** variable to receive the size of the buffer, in bytes, pointed to by the *Buffer* parameter.

ImageWidth: A pointer to a **DWORD** variable to receive the width, in pixels, of the fax image.

ImageHeight: A pointer to a **DWORD** variable to receive the height, in pixels, of the fax image.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x0000000D ERROR_INVALID_DATA	The data is invalid.

Return value/code	Description
0x0000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.49 FAX_GetPersonalCoverPagesOption (Opnum 40)

The **FAX_GetPersonalCoverPagesOption (Opnum 40)** method is called by the client to retrieve information about the supported personal cover-page options.

In response, the server MUST return the personal cover pages option that is supported by the server.

```
error_status_t FAX_GetPersonalCoverPagesOption(
  [in] handle_t hFaxHandle,
  [out, ref] LPBOOL lpbPersonalCPAllowed
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpbPersonalCPAllowed: A pointer to a **BOOL** that receives the personal cover-pages option. If **TRUE**, the server allows sending personal cover pages. Otherwise, the server does not allow personal cover pages.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.50 FAX_GetPersonalProfileInfo (Opnum 31)

The **FAX_GetPersonalProfileInfo (Opnum 31)** method is called by the client to retrieve information about the personal profile of a user from the specified fax message that is present in the described message folder. The Folder value MUST be one of the enumerations defined by **FAX_ENUM_MESSAGE_FOLDER (section 2.2.1)**. The ProfType value must be one of the enumerations that are defined by **FAX_ENUM_PERSONAL_PROF_TYPES (section 2.2.3)**.

In response, the server MUST validate that the message ID is for a valid message. The server MUST validate that the client has the access to read the message. On success, the server MUST return the profile information about the sender or recipient in *Buffer* along with the size.

The client SHOULD free the returned buffer.

```
error_status_t FAX_GetPersonalProfileInfo(
  [in] handle_t hBinding,
  [in] DWORDLONG dwlMessageId,
  [in] FAX_ENUM_MESSAGE_FOLDER dwFolder,
  [in] FAX_ENUM_PERSONAL_PROF_TYPES ProfType,
  [out, size_is(, *BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwlMessageId: A **DWORDLONG** that contains the message identifier for which the sender's **FAX_PERSONAL_PROFILE (section 2.2.33)** structure is retrieved.

dwFolder: A **FAX_ENUM_MESSAGE_FOLDER** (section 2.2.1) indicating the location of the folder in which to search for the message.

ProfType: A **FAX_ENUM_PERSONAL_PROF_TYPES** (section 2.2.3) indicating whether to retrieve sender or recipient information.

Buffer: A pointer to a **FAX_PERSONAL_PROFILE** (section 2.2.33) structure in which to place recipient job information.

BufferSize: A pointer to a **DWORD** variable to receive the buffer size.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000000D ERROR_INVALID_DATA	The data is invalid.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00001B61 FAX_ERR_MESSAGE_NOT_FOUND	The fax server cannot complete the operation because the number of active fax devices allowed for this version of Windows was exceeded.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.51 FAX_GetPort (Opnum 11)

The **FAX_GetPort (Opnum 11)** method is called by the client to retrieve port status information for a requested port at the server. The device ID passed in SHOULD be obtained from **FAX_EnumPorts (Opnum 11) (section 3.1.4.28)**. This is an extended version of **FAX_GetPort (Opnum 11)** (section 3.1.4.51).

The server MUST validate that the user has the access to get port status information. The *PortBuffer* parameter must not be **NULL**. The *FaxPortHandle* parameter MUST be returned by the **Fax_OpenPort (section 3.1.4.65)** method. On success, the server MUST return information for a specified fax port to a fax client application in *PortBuffer*.

The client SHOULD free the returned buffer.

```
error_status_t FAX_GetPort(
  [in] RPC_FAX_PORT_HANDLE FaxPortHandle,
  [out, size_is(,*BufferSize)] LPBYTE* PortBuffer,
  [out, ref] LPDWORD BufferSize
);
```

FaxPortHandle: A remote procedure call (RPC) context handle that references a specified fax port.

PortBuffer: A pointer to the address of a buffer to receive a **FAX_PORT_INFO (section 2.2.5)** structure. The structure describes one fax port.

BufferSize: A variable to return the size, in bytes, of the port buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x0000000D ERROR_INVALID_DATA	The data is invalid.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.52 FAX_GetPortEx (Opnum 46)

The **FAX_GetPortEx (Opnum 46)** method is called by the client to retrieve port status information for a requested port at the server. The device ID that is passed in should be obtained from **FAX_EnumPorts (section 3.1.4.28)**. This method is an extended version of **FAX_GetPort (section 3.1.4.51)**.

The server MUST validate that the user has the access to obtain port status information. The server MUST validate that *dwDeviceId* is for a valid device. The *Buffer* parameter MUST not be **NULL**.

On success, the server MUST return information about the specified fax port in *Buffer*.

The client SHOULD free the returned buffer.

```
error_status_t FAX_GetPortEx(
  [in] handle_t hFaxHandle,
  [in] DWORD dwDeviceId,
  [out, size_is(, *BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwDeviceId: A **DWORD** that indicates a unique identifier that distinguishes the device. Zero is an invalid value.

Buffer: A pointer to a buffer to hold a **FAX_PORT_INFO_EXW (section 2.2.34)** object.

BufferSize: A pointer to a **DWORD** in which to return the size, in bytes, of the buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x0000000D ERROR_INVALID_DATA	The data is invalid.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.53 FAX_GetQueueStates (Opnum 32)

The **FAX_GetQueueStates (Opnum 32)** method is called by the client to retrieve the state of the fax outgoing queue at the server.

The *pdwQueueStates* parameter MUST not be **NULL**. On success, the server MUST return the state information about the fax service.

```
error_status_t FAX_GetQueueStates(
  [in] handle_t hFaxHandle,
  [out] LPDWORD pdwQueueStates
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

pdwQueueStates: A pointer to a **DWORD** value that receives state information about the fax server queue. If this value is zero, both the incoming and outgoing queues are unblocked. Otherwise, this value is a combination of one or more of the following values.

Value	Meaning
0x00000000	Both the incoming and outgoing queues are unblocked.
FAX_INCOMING_BLOCKED 0x00000001	The fax service will not receive new incoming faxes.
FAX_OUTBOX_BLOCKED 0x00000002	The fax service will reject submissions of new outgoing faxes to its queue.
FAX_OUTBOX_PAUSED 0x00000004	The fax service will not dequeue and execute outgoing fax jobs from its queue.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.54 FAX_GetReceiptsConfiguration (Opnum 34)

The **FAX_GetReceiptsConfiguration (Opnum 34)** method is called by the client. In response, the server returns the receipts configuration information of the fax server.

```
error_status_t FAX_GetReceiptsConfiguration(
  [in] handle_t hFaxHandle,
  [out, size_is(, *BufferSize)] LPBYTE* Buffer,
  [out, ref] LPDWORD BufferSize
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Buffer: A pointer to a **FAX_RECEIPTS_CONFIGW** object, as defined in section **FAX_RECEIPTS_CONFIGW** (section 2.2.35).

BufferSize: A pointer to a **DWORD** in which to return the size, in bytes, of the buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.55 FAX_GetReceiptsOptions (Opnum 36)

The **FAX_GetReceiptsOptions (Opnum 36)** method is called by the client to retrieve the supported receipt options on the server.

The server MUST validate that the user has the access to retrieve the receipt options. On success, the server MUST return the receipt options that are supported by the server.

```
error_status_t FAX_GetReceiptsOptions(
  [in] handle_t hFaxHandle,
  [out, ref] LPDWORD lpdwReceiptsOptions
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpdwReceiptsOptions: A pointer to the **DWORD** that receives the options.

Value	Meaning
DRT_EMAIL 1	Allow sending the receipt by e-mail. The e-mail address is the e-mail address of the sender.
DRT_MSGBOX 4	Allow notification on the transmission result by using a message box to the sending machine of the user. **Note** The DRT_MSGBOX delivery method is not supported in Windows Server 2008 and later products.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.56 FAX_GetRecipientsLimit (Opnum 84)

The **FAX_GetRecipientsLimit (Opnum 84)** method is called by the client to retrieve information about the recipients limit of a single broadcast job.

The server MUST validate that the client has access to retrieve the recipients limit. On success, the server MUST return the maximum number of recipients to which a fax can be sent.<20>

```
error_status_t FAX_GetRecipientsLimit(
  [in] handle_t hBinding,
  [out, ref] LPDWORD lpdwRecipientsLimit
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpdwRecipientsLimit: A pointer to a **DWORD** value. Set to the maximum number of recipients to which a fax can be sent.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.57 FAX_GetRoutingInfo (Opnum 15)

The **FAX_GetRoutingInfo (Opnum 15)** method is called by the client to retrieve information about a specified routing method that is identified by the passed-in GUID.

The server MUST validate that the client has the access to retrieve information about a routing method. The *RoutingGuid* and *RoutingInfoBuffer* parameters MUST not be **NULL**. The server MUST validate that the *RoutingGuid* is for a valid routing method. On success, the server MUST return the routing information for a fax routing method that is associated with a specific fax device in *RoutingInfoBuffer*.

The client SHOULD free the returned buffer.

```
error_status_t FAX_GetRoutingInfo(
  [in] RPC_FAX_PORT_HANDLE FaxPortHandle,
  [in, string, unique] LPCWSTR RoutingGuid,
  [out, size_is(, *RoutingInfoBufferSize)]
    LPBYTE* RoutingInfoBuffer,
  [out, ref] LPDWORD RoutingInfoBufferSize
);
```

FaxPortHandle: A remote procedure call (RPC) context handle that references a specified fax port.

RoutingGuid: A pointer to a constant null-terminated Unicode string that specifies the globally unique identifier (GUID) that uniquely identifies the fax routing method for which to obtain the routing information. This string must be in the form "{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}". Fax routing methods are defined by a fax routing extension and the method is identified by a GUID. For more information about routing methods, see [MSDN-FRM].

The Microsoft fax routing provider defines the following routing methods.

Value	Meaning
REGVAL_RM_EMAIL_GUID "{6bbf7bfe-9af2-11d0-abf7-00c04fd91a4e}"	Get the information for routing faxes to send by e-mail. The *RoutingInfoBuffer* parameter is a pointer to a null-terminated Unicode string that contains the e-mail address that faxes are routed to.
REGVAL_RM_FOLDER_GUID "{92041a90-9af2-11d0-abf7-00c04fd91a4e}"	Get the information for routing faxes to a folder. The *RoutingInfoBuffer* parameter is a pointer to a null-terminated Unicode string that contains the path of the folder that faxes

Value	Meaning
	are routed to.
REGVAL_RM_PRINTING_GUID "{aec1b37c-9af2-11d0-abf7-00c04fd91a4e}"	Get the information for routing faxes to a printer. The *RoutingInfoBuffer* parameter is a pointer to a null-terminated Unicode string that contains the path and name of the printer that faxes are routed to.
REGVAL_RM_FLAGS_GUID "{aacc65ec-0091-40d6-a6f3-a2ed6057e1fa}"	Get the routing flags. The *RoutingInfoBuffer* parameter is a **DWORD** that contains the routing flags. The Microsoft fax routing provider defines the following routing flags. If this value is zero, none of these flags are set. Otherwise, this can be a combination of one or more of the following values. **Value** / **Meaning** 0x00000001 — Route faxes to a printer. 0x00000002 — Route faxes to a folder. 0x00000004 — Route faxes to the inbox. This flag is not supported. 0x00000008 — Route faxes to an e-mail address.

RoutingInfoBuffer: A pointer to the address of a buffer that receives the fax routing information. The buffer format and contents depend on the routing method that is identified by the *RoutingGuid* parameter.

RoutingInfoBufferSize: A pointer to a **DWORD** variable that receives the size, in bytes, of the *RoutingInfoBuffer* buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000001 ERROR_INVALID_FUNCTION	The function is incorrect.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x0000000D ERROR_INVALID_DATA	The data is invalid.
0x00000057	The parameter is incorrect.

Return value/code	Description
ERROR_INVALID_PARAMETER	

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.58 FAX_GetSecurity (Opnum 23)

The **FAX_GetSecurity (Opnum 23)** method is called by the client to retrieve information about the fax security descriptor from the fax server.

The server MUST validate that the client has access to retrieve security information. On success, the server MUST return the fax security descriptor from the fax server in *pSecurityDescriptor*. This function is no longer supported. **FAX_GetSecurityEx2 (Opnum 23) (section 3.1.4.60)** MUST be used instead of this function.

The client SHOULD free *pSecurityDescriptor*.

```
error_status_t FAX_GetSecurity(
  [in] handle_t hBinding,
  [out, size_is(, *lpdwBufferSize)]
    LPBYTE* pSecurityDescriptor,
  [out, ref] LPDWORD lpdwBufferSize
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

pSecurityDescriptor: A pointer to a **SECURITY_DESCRIPTOR** structure, as specified in [MS-DTYP] section 2.

lpdwBufferSize: A variable to return the size, in bytes, of the security descriptor buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000053A	The security descriptor structure is invalid.

Return value/code	Description
ERROR_INVALID_SECURITY_DESCR	

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.59 FAX_GetSecurityEx (Opnum 81)

The **FAX_GetSecurityEx (Opnum 81)** method is called by the client to retrieve information about the fax security descriptor from the fax server.<21>

The server MUST validate that the client has access to retrieve security information. On success, the server MUST return the fax security descriptor in *pSecurityDescriptor*. This function is no longer supported. **FAX_GetSecurityEx2 (Opnum 81) (section 3.1.4.60)** MUST be used instead of this function.

The client SHOULD free *pSecurityDescriptor*.

```
error_status_t FAX_GetSecurityEx(
  [in] handle_t hBinding,
  [in] SECURITY_INFORMATION SecurityInformation,
  [out, size_is(, *lpdwBufferSize)]
    LPBYTE* pSecurityDescriptor,
  [out, ref] LPDWORD lpdwBufferSize
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

SecurityInformation: Defines the wanted entries, indicated as a bitwise OR operator, in the security descriptor to return. For more information, see the **SECURITY_INFORMATION** data type.

pSecurityDescriptor: A pointer to a **SECURITY_DESCRIPTOR** structure, as specified in section 2 of [MS-DTYP].

lpdwBufferSize: A pointer to a **DWORD** value that indicates the size of the *pSecurityDescriptor* buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008	Not enough storage is available to process this command.

Return value/code	Description
ERROR_NOT_ENOUGH_MEMORY	
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000053A ERROR_INVALID_SECURITY_DESCR	The security descriptor structure is invalid.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.60 FAX_GetSecurityEx2 (Opnum 99)

The **FAX_GetSecurityEx2 (Opnum 99)** method is called by the client to retrieve information about the fax security descriptor from the fax server.

The server MUST validate that the client has the access to retrieve security information. The *pSecurityDescriptor* parameter must not be **NULL**. On success, the server MUST return the fax security descriptor from the fax server in *pSecurityDescriptor*.

The client SHOULD free the returned buffer.

```
error_status_t FAX_GetSecurityEx2(
  [in] handle_t hBinding,
  [in] SECURITY_INFORMATION SecurityInformation,
  [out, size_is(,*lpdwBufferSize)]
    LPBYTE* pSecurityDescriptor,
  [out, ref] LPDWORD lpdwBufferSize
);
```

hBinding: The handle that the client remote procedure call (RPC) layer provides when the RPC call is made.

SecurityInformation: Defines the desired entries, which are indicated as a bitwise OR operation, in the security descriptor to return.

For more information, see the description of the **SECURITY_INFORMATION** bit flags.

pSecurityDescriptor: A pointer to a **SECURITY_DESCRIPTOR** structure, as specified in section 2 of [MS-DTYP].

lpdwBufferSize: A pointer to a **DWORD** value that indicates the size, in bytes, of the *pSecurityDescriptor* buffer.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below:

Return value/code	Description
0x00000000	Method is successful.

Return value/code	Description
ERROR_SUCCESS	
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000053A ERROR_INVALID_SECURITY_DESCR	The security descriptor structure is invalid.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol [MS-RPCE].

3.1.4.61 FAX_GetServerActivity (Opnum 76)

The fax client application calls the **FAX_GetServerActivity (Opnum 76)** method to retrieve the status of the fax server queue activity and event log reports.

The client MUST allocate memory for the pServerActivity argument. It MUST also set the **dwSizeOfStruct** field of **FAX_SERVER_ACTIVITY (section 2.2.15)** to the size of the **FAX_SERVER_ACTIVITY** (section 2.2.15) structure.

In response, the server MUST validate that the client has access to the server logs. On success, the server MUST return information about its activity and event logs.

```
error_status_t FAX_GetServerActivity(
  [in] handle_t hFaxHandle,
  [in, out, ref] PFAX_SERVER_ACTIVITY pServerActivity
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

pServerActivity: A pointer to a **FAX_SERVER_ACTIVITY** (section 2.2.15) object.

Return Values: This method MUST return zero (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057	The parameter is incorrect.

Return value/code	Description
ERROR_INVALID_PARAMETER	

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.62 FAX_GetServerSKU (Opnum 85)

The **FAX_GetServerSKU (Opnum 85)** method is called by the client. In response, the server returns the stock-keeping unit (SKU) of the fax server operating system.

The server MUST check if the client has permissions to know the server SKU type. On success, the server MUST return its SKU type.

```
error_status_t FAX_GetServerSKU(
  [in] handle_t hbinding,
  [out, ref] PRODUCT_SKU_TYPE* pServerSKU
);
```

hbinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

pServerSKU: A pointer to a **PRODUCT_SKU_TYPE (section 2.2.61)** enumeration that receives the fax server SKU.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.63 FAX_GetServicePrinters (Opnum 0)

The fax client application calls the **FAX_GetServicePrinters (Opnum 0)** method to obtain a list of printers that the fax service is aware of.

In response, the server MUST validate if the client has access to enumerate the printers that are associated with the server. On success, the server returns information about all the printers it has in lpBuffer. It MUST also return the size of this information and the number of printers for which it enumerated the information successfully.

The client SHOULD free the returned buffer.

```
error_status_t FAX_GetServicePrinters(
  [in] handle_t hBinding,
  [out, size_is(,*lpdwBufferSize)]
    LPBYTE* lpBuffer,
  [out, ref] LPDWORD lpdwBufferSize,
  [out, ref] LPDWORD lpdwPrintersReturned
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpBuffer: A pointer to a buffer containing an array of **FAX_PRINTER_INFO (section 2.2.32)** structures.

lpdwBufferSize: A pointer to a **DWORD** value containing the size, in bytes, of the buffer.

lpdwPrintersReturned: A pointer to a **DWORD** value indicating the number of the printers in the buffer.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.64 FAX_GetVersion (Opnum 37)

The fax client application calls the **FAX_GetVersion (Opnum 37)** method to obtain the version of the fax server it is connected to.

In response, the server MUST check if the client has permissions to know the fax version. On success, the server MUST return its version.

```
error_status_t FAX_GetVersion(
  [in] handle_t hFaxHandle,
  [in, out] PFAX_VERSION pVersion
);
```

hFaxHandle: The handle provided by the client RPC layer when the RPC call is made.

pVersion: A pointer to a **FAX_VERSION (section 2.2.17)** object.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.65 FAX_OpenPort (Opnum 2)

The **FAX_OpenPort (Opnum 2)** method is called by the client. In response, the server opens a fax port for subsequent use in other fax methods and returns a fax port handle for use by the fax client application.

In response, the server MUST validate if the client has access to open the specified fax port. The server MUST validate that the DeviceId argument that is passed by the client is for a valid device. If the Flags argument specifies PORT_OPEN_MODIFY, the server MUST also confirm that no other client has opened the port for modification. To indicate success, the server MUST return a new port handle to the client.

```
error_status_t FAX_OpenPort(
  [in] handle_t hBinding,
  [in] DWORD DeviceId,
  [in] DWORD Flags,
  [out] PRPC_FAX_PORT_HANDLE FaxPortHandle
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

DeviceId: A **DWORD** variable that is the permanent line identifier for the receiving device. The client should have called the **FAX_EnumPorts (section 3.1.4.28)** method to retrieve a valid value for this parameter.

Flags: A **DWORD** variable that contains a set of bit flags defining the access level for the port.

Value	Meaning
PORT_OPEN_QUERY 1	The port access level that is required to obtain a fax port handle. This access level is also required to call the **FAX_GetPort (section 3.1.4.51)** method to query fax port information.
PORT_OPEN_MODIFY	The port access level that is required to change the configuration of a fax

Value	Meaning
2	port with a call to the **FAX_SetPort (section 3.1.4.88)** method. This access level also includes the access rights that are associated with the PORT_OPEN_QUERY access level.

FaxPortHandle: A pointer to a variable that receives a fax port handle (as defined in section 2.2.49) required on subsequent calls by other fax client methods. This method should return a **NULL** handle to indicate an error.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000014 ERROR_BAD_UNIT	The system cannot find the specified device.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00000649 ERROR_INVALID_HANDLE	The handle is in an invalid state.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.66 FAX_ReadFile (Opnum 71)

The fax client application calls the **FAX_ReadFile (Opnum 71)** method to copy a file from the server (in chunks).

In response, the server MUST validate that the hCopy context handle refers to a valid copy handle returned by **FAX_StartCopyMessageFromServer (Opnum 71) (section 3.1.4.96)** and for which FAX_EndCopy has not been called. To indicate success, the server MUST copy the contents of the specified message into the buffer and MUST return the buffer to the client. The server MUST also return the number of bytes it wrote successfully to the buffer. The server MUST NOT write more than dwMaxDataSize bytes to the buffer.

The client SHOULD free the returned buffer.

```
error_status_t FAX_ReadFile(
  [in, ref] RPC_FAX_COPY_HANDLE hCopy,
  [in] DWORD dwMaxDataSize,
  [out, ref, size_is(*lpdwDataSize)]
    LPBYTE lpbData,
  [in, out, ref] LPRANGED_DWORD lpdwDataSize
);
```

hCopy: An RPC context handle returned by **FAX_StartCopyMessageFromServer** (section 3.1.4.96).

dwMaxDataSize: A **DWORD** value that indicates the maximum size, in bytes, of data to send in one segment.

lpbData: A pointer to the buffer in which to place the data.

lpdwDataSize: A pointer to a **DWORD** in which to return the size, in bytes, of the data that is sent in this segment.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00000649 ERROR_INVALID_HANDLE	The handle is in an invalid state.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.67 FAX_ReAssignMessage (Opnum 102)

The fax client application calls the **FAX_ReAssignMessage (Opnum 102)** method to assign the specified fax message to a set of users.

The client MUST specify the recipients for a reassigned message in a comma (;) separated format. In response, the server MUST validate whether it supports reassign and whether the client has reassign permissions. The server MUST also validate if the message that is specified by the dwlMessageId argument refers to a valid message on the server and the client has access to this message. The server MUST validate whether the recipient numbers for each of the recipients that are listed in pReAssignInfo structure.

```
error_status_t FAX_ReAssignMessage(
  [in] handle_t hBinding,
  [in] DWORDLONG dwlMessageId,
  [in, ref] PFAX_REASSIGN_INFO pReAssignInfo
);
```

hBinding: The fax server handle that is returned by a call to the **Fax_ConnectFaxServer (section 3.1.4.10)** method. This is a context handle.

dwlMessageId: A **DWORDLONG** value that specifies the identifier of the fax message to assign or reassign.

pReAssignInfo: A pointer to a **FAX_REASSIGN_INFO (section 2.2.13)** structure that contains assignment or reassignment information.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000002 ERROR_FILE_NOT_FOUND	The system cannot find the specified file.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x000010DD ERROR_INVALID_OPERATION	The operation identifier is not valid.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.68 FAX_RefreshArchive (Opnum 82)

A fax client application calls the **FAX_RefreshArchive (Opnum 82)** method to notify the server that the archive folder has been changed and should be refreshed.<22>

In response, the server MUST validate that the client has access to the specified folder. On success, the server MUST update its data about the specified folder.

```
error_status_t FAX_RefreshArchive(
  [in] handle_t hFaxHandle,
  [in] FAX_ENUM_MESSAGE_FOLDER Folder
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Folder: A value indicating the archive folder to refresh. For more information, see **FAX_ENUM_MESSAGE_FOLDER (section 2.2.1)**.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.

Return value/code	Description
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x000010DD ERROR_INVALID_OPERATION	The operation identifier is not valid.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.69 FAX_RegisterServiceProviderEx (Opnum 60)

The fax client application calls the **FAX_RegisterServiceProviderEx (Opnum 60)** method to register a fax service provider (FSP) with the Fax Service. Registration takes place after the Fax Service restarts. Only an administrator can register an FSP.

In response, the server MUST validate that the client has access to register an FSP. The server MUST also validate that the guidlpctstrGUID is not a duplicate because it MUST NOT register duplicate FSPs.

```
error_status_t FAX_RegisterServiceProviderEx(
  [in] handle_t hFaxHandle,
  [in, string, ref] LPCWSTR lpctstrGUID,
  [in, string, ref] LPCWSTR lpctstrFriendlyName,
  [in, string, ref] LPCWSTR lpctstrImageName,
  [in, string, ref] LPCWSTR lpctstrTspName,
  [in] DWORD dwFSPIVersion,
  [in] DWORD dwCapabilities
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpctstrGUID: A pointer to a constant null-terminated character string that contains a valid string representation of the globally unique identifier of the FSP.

lpctstrFriendlyName: A pointer to a constant null-terminated character string to associate with the FSP execution component. This is the FSP friendly name, which is suitable for display. This value cannot exceed 258 (MAX_FAX_STRING_LEN) characters.

lpctstrImageName: A pointer to a constant null-terminated character string that specifies the full path and file name for the FSP execution component. In Windows, this is a path to the FSP DLL. This value cannot exceed 258 (MAX_FAX_STRING_LEN) characters.

lpctstrTspName: A pointer to a constant null-terminated character string that specifies the name of the telephony service provider that is associated with the devices for the FSP. This parameter should be set to NULL if the FSP does not use a telephony service provider. This value cannot exceed 258 (MAX_FAX_STRING_LEN) characters. This value must be unique across all registered FSPs.

dwFSPIVersion: A **DWORD** value that specifies the API version of the FSP interface. A client only provides a value of 0x00010000 (FSPI_API_VERSION_1) in this parameter.

dwCapabilities: A **DWORD** value that specifies the capabilities of the extended FSP. This parameter is always set to 0.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.
0x000000B7 ERROR_ALREADY_EXISTS	You cannot create a file when that file already exists.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.70 FAX_RemoveMessage (Opnum 67)

The fax client application calls the **FAX_RemoveMessage (Opnum 67)** method to remove a message from a specific fax archive folder.

In response, the server MUST validate if the client has access to remove a message from the server. The server MUST also validate whether the dwlMessageId argument refers to a valid message in the folder that is specified by the *Folder* parameter and whether the client has access to this message.

```
error_status_t FAX_RemoveMessage(
  [in] handle_t hFaxHandle,
  [in] DWORDLONG dwlMessageId,
  [in] FAX_ENUM_MESSAGE_FOLDER Folder
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwlMessageId: A **DWORD** value identifying the fax message to remove from the archive.

Folder: The type of the archive where the message resides. FAX_MESSAGE_FOLDER_QUEUE is an invalid value for this parameter.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000000E ERROR_OUTOFMEMORY	Not enough storage is available to complete this operation.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00001B60 FAX_ERR_FILE_ACCESS_DENIED	The fax server cannot find the job or message by its identifier.
0x00001B61 FAX_ERR_MESSAGE_NOT_FOUND	The fax server cannot complete the operation because the number of active fax devices that are allowed for this version of Windows was exceeded.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.71 FAX_RemoveOutboundGroup (Opnum 53)

The fax client application calls the **FAX_RemoveOutboundGroup (Opnum 53)** method to remove an existing outbound routing group from the fax server.

In response, the server MUST validate that the lpwstrGroupName does not specify the special routing group called "All Devices", because the same cannot be removed. It MUST validate that the server is NOT running on a desktop SKU. The client MUST have access to manage configuration on the server. The server MUST also confirm that the group is NOT being used in a rule.

```
error_status_t FAX_RemoveOutboundGroup(
  [in] handle_t hFaxHandle,
  [in, string, ref] LPCWSTR lpwstrGroupName
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpwstrGroupName: A pointer to a null-terminated string that uniquely identifies an existing group name. The group name is expected to be case-insensitive.

Return Values: This method MUST return 0 (ERROR_SUCCESS) to indicate success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.

Return value/code	Description
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.
0x000010DD ERROR_INVALID_OPERATION	The operation identifier is not valid.
0x00001B5C FAX_ERR_GROUP_IN_USE	The fax server cannot remove an outbound routing group because it is in use by one or more outbound routing rules.
0x00001B63 FAX_ERR_NOT_SUPPORTED_ON_THIS_SKU	The fax server API version does not support the requested operation.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.72 FAX_RemoveOutboundRule (Opnum 57)

The **FAX_RemoveOutboundRule (Opnum 57)** method removes an existing outbound routing rule from the rules map. The default outbound routing rule cannot be removed.

In response, the server MUST validate that the client has access to manage configuration. The country/region code MUST NOT be 0, because the same indicates it to correspond to any country/region. The server MUST make sure it is NOT running on a desktop SKU.

```
error_status_t FAX_RemoveOutboundRule(
  [in] handle_t hFaxHandle,
  [in] DWORD dwAreaCode,
  [in] DWORD dwCountryCode
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwAreaCode: The area code of the rule. If this value is zero, a special 'All other areas' value should be used. The *dwAreaCode* and *dwCountryCode* parameters are a unique key.

dwCountryCode: The country code of the rule. If this value is zero, a special 'All other countries' value should be used. The *dwAreaCode* and *dwCountryCode* parameters are a unique key.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.
0x000010DD ERROR_INVALID_OPERATION	The operation identifier is not valid.
0x00001B5D FAX_ERR_RULE_NOT_FOUND	The fax server failed to locate an outbound routing rule by country/region code and area code.
0x00001B63 FAX_ERR_NOT_SUPPORTED_ON_THIS_SKU	The fax server API version does not support the requested operation.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.73 FAX_SendDocumentEx (Opnum 27)

The **FAX_SendDocumentEx (Opnum 27)** method is called by the client. In response, the server sends the specified fax job.

To succeed, the Submit method requires that at least one recipient, and either a cover page or a **fax body** are present.<23>

You can only use this method if the server (remote or local) is installed as a network printer on the local computer.

```
error_status_t FAX_SendDocumentEx(
  [in] handle_t hBinding,
  [in, string, unique] LPCWSTR lpcwstrFileName,
  [in] LPCFAX_COVERPAGE_INFO_EXW lpcCoverPageInfo,
  [in] LPCFAX_PERSONAL_PROFILEW lpcSenderProfile,
  [in, range (0,FAX_MAX_RECIPIENTS)]
    DWORD dwNumRecipients,
  [in, size_is(dwNumRecipients)]
    LPCFAX_PERSONAL_PROFILEW lpcRecipientList,
  [in] LPCFAX_JOB_PARAM_EXW lpJobParams,
  [in, out, unique] LPDWORD lpdwJobId,
  [out] PDWORDLONG lpdwlMessageId,
  [out, size_is(dwNumRecipients)]
    PDWORDLONG lpdwlRecipientMessageIds
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpcwstrFileName: A pointer to a null-terminated string that contains the name of the file, without path information, of the **body** of the fax in **TIFF**. The file name is retrieved from a previous call to **FAX_StartCopyToServer (section 3.1.4.97)** that copies the body file to the server queue.

lpcCoverPageInfo: A pointer to a **FAX_COVERPAGE_INFO_EX (section 2.2.9)** structure that contains the cover-page information. This is never **NULL**. If no cover-page information is available, the **lptstrCoverPageFileName** member of the structure is **NULL**. If cover-page information is specified, it is expected that it point to a cover-page template file on the server.

lpcSenderProfile: A pointer to a **FAX_PERSONAL_PROFILE (section 2.2.33)** structure that contains the personal details of the fax sender.

dwNumRecipients: A DWORD that contains the number of recipients of the fax.

lpcRecipientList: A pointer to an array **FAX_PERSONAL_PROFILE** (section 2.2.33) structure that contains personal details about the recipients of the fax. The **dwNumRecipients** member specifies the number of elements in this array.

lpJobParams: A pointer to a **FAX_JOB_PARAM_EX (section 2.2.10)** structure that contains the information necessary for the fax server to send the fax transmission.

lpdwJobId: A pointer to a **DWORD** that returns the job session ID of the recipient (only one recipient). This parameter is used for backward compatibility with FAX_SendDocument. If this parameter is **NULL**, you may ignore it.

lpdwlMessageId: A pointer to a **DWORDLONG** that returns the unique identifier of the job that represents the entire transmission.

A pointer to an array of **DWORDLONG**s in which you return the unique message identifier for each recipient. The number of elements in this array should be at least equal to the value specified in the **dwNumRecipients** member. The elements in the array are expected to be ordered in the same order as the elements of the *lpcRecipientList* array.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
0x00000013 ERROR_WRITE_PROTECT	The media is write-protected.
0x00000032 ERROR_NOT_SUPPORTED	The request is not supported.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Return value/code	Description
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.
0x0000065E ERROR_UNSUPPORTED_TYPE	Data of this type is not supported.
0x0000070C ERROR_INVALID_DATA	The specified data type is invalid.
0x00001B63 FAX_ERR_NOT_SUPPORTED_ON_THIS_SKU	The fax server API version does not support the requested operation.
0x00001B65 FAX_ERR_RECIPIENTS_LIMIT	The limit on the number of recipients for a single fax broadcast was reached.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.74 FAX_SetActivityLoggingConfiguration (Opnum 44)

The fax client application calls the **FAX_SetActivityLoggingConfiguration (Opnum 44)** method to set options for activity logging. This includes setting whether entries for incoming and outgoing faxes should be logged and the location of the log file.

In response, the server MUST check that the client has access to manage server configuration. It MUST validate the logging parameters, including the path that is specified to the log file.

```
error_status_t FAX_SetActivityLoggingConfiguration(
  [in] handle_t hFaxHandle,
  [in, ref] const PFAX_ACTIVITY_LOGGING_CONFIGW pActivLogCfg
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

pActivLogCfg: A pointer to a **FAX_ACTIVITY_LOGGING_CONFIGW (section 2.2.19)** object.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Return value/code	Description
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.
0x000003F7 ERROR_REGISTRY_CORRUPT	The registry is corrupted. The structure of one of the files that contain registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.
0x00001B60 FAX_ERR_FILE_ACCESS_DENIED	The fax server cannot find the job or message by its identifier.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.75 FAX_SetArchiveConfiguration (Opnum 42)

The fax client application calls the **FAX_SetArchiveConfiguration (Opnum 42)** method to set the archive configuration for a specific fax folder on the fax server.

In response, the server MUST validate that the client has access to manage server configuration. On success, the server MUST set the specified configuration and return success.<24>

```
error_status_t FAX_SetArchiveConfiguration(
  [in] handle_t hFaxHandle,
  [in] FAX_ENUM_MESSAGE_FOLDER Folder,
  [in, ref] const PFAX_ARCHIVE_CONFIGW pArchiveCfg
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Folder: The archive location. The client expects either FAX_MESSAGE_FOLDER_INBOX or FAX_MESSAGE_FOLDER_SENTITEMS as values.

pArchiveCfg: A pointer to a **FAX_ARCHIVE_CONFIGW (section 2.2.20)** object.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.

Return value/code	Description
0x000003F7 ERROR_REGISTRY_CORRUPT	The registry is corrupted. The structure of one of the files that contain registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.
0x00001B60 FAX_ERR_FILE_ACCESS_DENIED	The fax server cannot find the job or message by its identifier.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.76 FAX_SetConfiguration (Opnum 20)

The fax client application calls the **FAX_SetConfiguration (Opnum 20)** method to change the fax configuration on the fax server. The SizeOfStruct in the FaxConfig argument MUST be set to a value of sizeof(FAX_CONFIGURATIONW).

In response, the server MUST validate that the client has access to manage configuration on the server. On success, it sets the specified configuration parameters and returns success.

```
error_status_t FAX_SetConfiguration(
  [in] handle_t hBinding,
  [in] const FAX_CONFIGURATIONW* FaxConfig
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

FaxConfig: A pointer to a **FAX_CONFIGURATIONW** structure (as defined in section 2.2.21).

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.
0x000003F7 ERROR_REGISTRY_CORRUPT	The registry is corrupted. The structure of one of the files that contain registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.

Return value/code	Description
0x00001B60 FAX_ERR_FILE_ACCESS_DENIED	The fax server cannot find the job or message by its identifier.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.77 FAX_SetConfigWizardUsed (Opnum 77)

The **FAX_SetConfigWizardUsed** method is called by the client. In response, the server sets a value in the registry indicating that the configuration wizard was used. The server MUST validate that the client has access to manage configuration information on the server.

```
error_status_t FAX_SetConfigWizardUsed(
  [in] handle_t hFaxHandle,
  [in] BOOL bConfigWizardUsed
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

bConfigWizardUsed: A Boolean value indicating whether the fax configuration wizard was used.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which is listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.

The registry entry set by this method is:
HKEY_LOCAL_MACHINE\Software\Microsoft\Fax\CfgWzdrDevice

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.78 FAX_SetDeviceOrderInGroup (Opnum 55)

The **FAX_SetDeviceOrderInGroup** method is called by the client. In response, the server sets the order of a single device in a group of outbound routing devices.

The order is the 1-based location of the device in the group. The value of 1 indicates the device is ordered first in the group. The order of devices in the group determines the order in which they are used to send outgoing faxes when the group is selected by an outbound routing rule.

The server MUST validate that the group name length is within allowed limits and that it is NOT running on a desktop SKU. It MUST validate that the client has access to manage configuration on the server. It MUST validate that dwNewOrder is within the limits of the specified group.

```
error_status_t FAX_SetDeviceOrderInGroup(
  [in] handle_t hFaxHandle,
  [in, string, ref] LPCWSTR lpwstrGroupName,
  [in] DWORD dwDeviceId,
  [in] DWORD dwNewOrder
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpwstrGroupName: A pointer to a null-terminated string that uniquely identifies a group. Group names should be case-insensitive.

dwDeviceId: A **DWORD** value specifying the identifier of the device in the group. The specified device must exist in the group.

dwNewOrder: A **DWORD** value specifying the new 1-based order of the device in the group. If there are N devices in the group, this value must be between 1 and N (inclusive). Other devices are moved up or down in the group to place the specified device in the specified order.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.
0x000003F7 ERROR_REGISTRY_CORRUPT	The registry is corrupted. The structure of one of the files containing registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.
0x00001B63 FAX_ERR_NOT_SUPPORTED_ON_THIS_SKU	The fax server API version does not support the requested operation.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol [MS-RPCE].

3.1.4.79 FAX_SetExtensionData (Opnum 50)

The fax client application calls the **FAX_SetExtensionData (Opnum 50)** method in order to write the extension data for a device.

In response, the server MUST validate that the client has access to manage configuration on the server. It also MUST validate that the lpcstrNameGUID is for a valid routing method.

```
error_status_t FAX_SetExtensionData(
  [in] handle_t hFaxHandle,
  [in, string] LPCWSTR lpcwstrComputerName,
  [in] DWORD dwDeviceId,
  [in, string] LPCWSTR lpctstrNameGUID,
  [in, ref, size_is(dwDataSize)]
    LPBYTE pData,
  [in, range (0,FAX_MAX_RPC_BUFFER)]
    DWORD dwDataSize
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpcwstrComputerName: A string that specifies the name of the computer that is calling the module.

dwDeviceId: A **DWORD** value of the unique device identifier. A value of zero indicates the caller wants to set a named data BLOB that is not associated with any specific device. This value can be used to store configurations that affect all the devices. For example, an Optical Character Recognition (OCR) routing extension might export several different routing methods that rely on the same OCR parameters. This routing extension should associate the OCR configuration with a non-specific device so that it would become global.

lpctstrNameGUID: A string GUID that identifies the data to set. This string must be in the form "{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}". If some data is already set for the specified globally unique identifier (GUID), it is expected that it is replaced with the new data that is pointed to by the *pData* parameter.

pData: A pointer to the data buffer to set. If this pointer is **NULL**, the data at the server is removed from persistent storage.

dwDataSize: A **DWORD** value that indicates the size, in bytes, of the *pData* buffer.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.80 FAX_SetGeneralConfiguration (Opnum 98)

The fax client application calls the **FAX_SetGeneralConfiguration (Opnum 98**) method to set the configuration options provided for the fax service. The **FAX_GENERAL_CONFIG (section 2.2.23)** structure MUST be serialized. The variable data fields, such as strings, MUST be filled with the offset to the string from the beginning of the buffer and not the actual address. In response, the server MUST validate that the client has access to manage configuration on the server. On success, the desired configuration information is set and success is returned.

```
error_status_t FAX_SetGeneralConfiguration(
  [in] handle_t hBinding,
  [in] DWORD level,
  [in, ref, size_is(BufferSize)]
    const LPBYTE Buffer,
  [in, range (0,FAX_MAX_RPC_BUFFER)]
    DWORD BufferSize
);
```

hBinding: The fax server handle that is returned by a call to the **Fax_ConnectFaxServer (section 3.1.4.10)** method. This is a context handle.

level: A **DWORD** value that indicates the type of structure to return in **Buffer**. This value must be set to 0.

Buffer: A pointer to a **FAX_GENERAL_CONFIG** (section 2.2.23) structure that contains the configuration information to set.

BufferSize: A pointer to a **DWORD** value that specifies the size, in bytes, of the buffer that is pointed to by the **Buffer** parameter.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

The FAX_GENERAL_CONFIG structure must be serialized. The variable data fields, such as strings, should be filled with the offset to the string from the beginning of the buffer and not the actual address.

3.1.4.81 FAX_SetGlobalRoutingInfo (Opnum 18)

The fax client application calls the **FAX_SetGlobalRoutingInfo (Opnum 18)** method to set global routing properties such as the routing method priority.

In response, the server MUST validate that the client has access to set the global routing information on the server. On success, the server MUST modify its fax routing method data, such as routing priority, that applies globally.

```
error_status_t FAX_SetGlobalRoutingInfo(
  [in] handle_t hBinding,
  [in] const FAX_GLOBAL_ROUTING_INFOW* RoutingInfo
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

RoutingInfo: A pointer to a buffer that contains a **FAX_GLOBAL_ROUTING_INFOW (section 2.2.24)** structure.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000000D ERROR_INVALID_DATA	The data is invalid.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.82 FAX_SetJob (Opnum 6)

The **FAX_SetJob** method is called by the client. In response, the server pauses, resumes, cancels, or restarts the specified fax job.

This function sets job status information for a requested job ID. If a single fax from a broadcast is canceled, the status of the job is set according to the parent job status. The canceled fax remains in the outgoing queue so that the status of all faxes can be viewed from the broadcast.

```
error_status_t FAX_SetJob(
  [in] handle_t hBinding,
  [in] DWORD JobId,
  [in] DWORD Command
```

```
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

JobId: A **DWORD** variable that uniquely identifies the fax job to modify.

Command: A **DWORD** variable that indicates the job command to perform.

Value	Meaning
JC_DELETE 1	Cancel the specified fax job. The job can be active or queued.
JC_PAUSE 2	Pause the specified queued fax job. If the fax job is active, the fax service pauses the job when it returns to the queued state.
JC_RESUME 3	Resume the paused fax job.
JC_RESTART 3	Restart the specified fax job.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.83 FAX_SetLoggingCategories (Opnum 22)

The **FAX_SetLoggingCategories** method is called by the client. In response, the server modifies the current logging categories for the fax server to which the client has connected. A logging category determines the errors or other events that the fax server records in the application event log.

```
error_status_t FAX_SetLoggingCategories(
  [in] handle_t hBinding,
  [in, unique, size_is(BufferSize)]
    const LPBYTE Buffer,
  [in, range (0,FAX_MAX_RPC_BUFFER)]
    DWORD BufferSize,
```

```
  [in] DWORD NumberCategories
);
```

hBinding: A handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Buffer: A pointer to an array of the **FAX_LOG_CATEGORY (section 2.2.8)** structure. Each structure contains the data to modify one logging category. The data includes the descriptive name of the logging category, the category number, and the current logging level for the category. For more information, see [MSDN-FSCAR].

BufferSize: A variable to return the size, in bytes, of the job information buffer. Must be a value between 0 and 1,048,576.

NumberCategories: A **DWORD** variable that contains the number of **FAX_LOG_CATEGORY** (section 2.2.8) structure items that the method passes in the *Buffer* parameter.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

The FAX_LOG_CATEGORY structure array must be serialized. For more information, see [MSDN-FAX_LOG_CATEGORY]. The variable data fields, such as strings, should be filled with the offset to the string from the beginning of the buffer and not the actual address.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.84 FAX_SetMessage (Opnum 103)

The fax client application calls the **FAX_SetMessage (Opnum 103)** method to set the specific message properties for the message identified by its ID.

In response, the server MUST validate if the client has access to set the message properties. The server MUST also confirm if the dwlMessageId specified by the client refers to a valid message and client has access to this message in the specified folder. On success, the server MUST set the specified message properties for the fax message.

```
error_status_t FAX_SetMessage(
  [in] handle_t hFaxHandle,
  [in] DWORDLONG dwlMessageId,
  [in] FAX_ENUM_MESSAGE_FOLDER Folder,
  [in, ref] PFAX_MESSAGE_PROPS lpMessageProps
```

```
);
```

hFaxHandle: The fax server handle that is returned by a call to the **Fax_ConnectFaxServer (section 3.1.4.10)** method. This is a context handle.

dwlMessageId: The unique ID number of the fax message.

Folder: Identifies the location of the fax message. The value in this parameter must come from the **FAX_ENUM_MESSAGE_FOLDER** enumeration. It can be set to the FAX_MESSAGE_FOLDER_INBOX or FAX_MESSAGE_FOLDER_SENTITEMS constant.

lpMessageProps: This MUST be a pointer to a **FAX_MESSAGE_PROPS (section 2.2.11)** structure. Contains the property settings for the fax message identified by dwlMessageId.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The structure pointed to by *lpMessageProps* contains invalid data, the *Folder* parameter has an invalid value, or the *dwlMessageId* parameter is zero.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.85 FAX_SetOutboundGroup (Opnum 52)

The fax client application calls the **FAX_SetOutboundGroup (Opnum 52)** method to set a new device list to an existing group.

In response, the server MUST validate if the client has access to set the outbound routing groups. It MUST validate if the dwSizeOfStruct field passed in pGroup is equal to the size of RPC_FAX_OUTBOUND_ROUTING_GROUPW structure. On success, the server MUST modify its outbound routing groups as specified by the client.

```
error_status_t FAX_SetOutboundGroup(
  [in] handle_t hFaxHandle,
  [in, ref] PRPC_FAX_OUTBOUND_ROUTING_GROUPW pGroup
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

pGroup: A pointer to a **RPC_FAX_OUTBOUND_ROUTING_GROUPW (section 2.2.30)** buffer to set.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.
0x000003F7 ERROR_REGISTRY_CORRUPT	The registry is corrupted. The structure of one of the files containing registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.
0x00001B63 FAX_ERR_NOT_SUPPORTED_ON_THIS_SKU	The fax server API version does not support the requested operation.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.86 FAX_SetOutboundRule (Opnum 58)

A fax client application uses the **FAX_SetOutboundRule (Opnum 58)** method to set the information about an individual fax outbound routing rule.

In response, the server MUST validate that the client has access to set an outbound routing rule. On success, the server MUST modify its outbound routing rule as specified by the client.

```
error_status_t FAX_SetOutboundRule(
  [in] handle_t hFaxHandle,
  [in, ref] RPC_FAX_OUTBOUND_ROUTING_RULEW* pRule
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

pRule: A pointer to an **RPC_FAX_OUTBOUND_ROUTING_RULEW (section 2.2.31)** buffer to set.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.
0x000003F7 ERROR_REGISTRY_CORRUPT	The registry is corrupted. The structure of one of the files containing registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.
0x00001B63 FAX_ERR_NOT_SUPPORTED_ON_THIS_SKU	The fax server API version does not support the requested operation.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.87 FAX_SetOutboxConfiguration (Opnum 39)

The fax client application calls the **FAX_SetOutboxConfiguration (Opnum 39)** method to set the current Outbox configuration such as the Discount Time.

In response, the server MUST validate if the client has access to set an outbound routing configuration. On success, the server MUST modify its outbound routing configuration as specified by the client.

```
error_status_t FAX_SetOutboxConfiguration(
  [in] handle_t hFaxHandle,
  [in, ref] const PFAX_OUTBOX_CONFIG pOutboxCfg
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

pOutboxCfg: A pointer to an **FAX_OUTBOX_CONFIG (section 2.2.12)** object containing configuration information.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.

Return value/code	Description
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x000003F7 ERROR_REGISTRY_CORRUPT	The registry is corrupted. The structure of one of the files containing registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.88 FAX_SetPort (Opnum 12)

A fax client application uses the **FAX_SetPort (Opnum 12)** method to set fax device information. The functions sets extension configuration properties that are stored at the device level, such as enabling or disabling sending and receiving, and the auto or manual answering of calls.

In response, the server MUST validate if the FaxPortHandle argument that is passed by the client refers to a port handle that is obtained by a call to FAX_OpenPort. This port MUST have been opened by using the PORT_OPEN_MODIFY value. The server MUST validate if the client has access to modify the properties of this port. On success, the server MUST modify the properties of the port as specified by the client.

```
error_status_t FAX_SetPort(
  [in] RPC_FAX_PORT_HANDLE FaxPortHandle,
  [in] FAX_PORT_INFO* PortInfo
);
```

FaxPortHandle: A remote procedure call (RPC) context handle that references a specified fax port.

PortInfo: A pointer to a **FAX_PORT_INFO (section 2.2.5)** structure. The structure contains data to modify the configuration of the specified fax port.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000000D ERROR_INVALID_DATA	The data is invalid.

Return value/code	Description
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00001B62 FAX_ERR_DEVICE_NUM_LIMIT_EXCEEDED	The fax server cannot complete the operation because it is not supported for this version of Windows.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.89 FAX_SetPortEx (Opnum 47)

A fax client application uses the **FAX_SetPortEx (Opnum 47)** method to set fax device information. The function sets extension configuration properties that are stored at the device level, such as enable or disable sending and receiving, and the auto or manual answering of calls.

In response, the server MUST validate if the client has access to the server. The server MUST validate that the dwDeviceId parameter that is specified by the client is for a valid device. On success, the server MUST modify the properties of the device as specified by the client.

```
error_status_t FAX_SetPortEx(
  [in] handle_t hFaxHandle,
  [in] DWORD dwDeviceId,
  [in, ref] const PFAX_PORT_INFO_EXW pPortInfo
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwDeviceId: A unique identifier that distinguishes the device. A zero value is invalid.

pPortInfo: A pointer to a buffer of type **FAX_PORT_INFO_EXW (section 2.2.34)**.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000000D ERROR_INVALID_DATA	The data is invalid.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00001B62 FAX_ERR_DEVICE_NUM_LIMIT_EXCEEDED	The fax server cannot complete the operation because it is not supported for this version of Windows.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.90 FAX_SetQueue (Opnum 33)

The fax client application calls the **FAX_SetQueue (Opnum 33)** method to change the state of the server queue. In response, the server MUST validate if the client has access to set the queue state of the server. On success, the server MUST set its queue state as specified by the client.

```
error_status_t FAX_SetQueue(
  [in] handle_t hFaxHandle,
  [in] const DWORD dwQueueStates
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwQueueStates: A pointer to a **DWORD** value that contains state information about the fax server queue. If this value is zero, both the incoming and outgoing queues are unblocked. Otherwise, this value is a combination of one or more of the following values.

Value	Meaning
0x00000000	Both the incoming and outgoing queues are unblocked.
FAX_INCOMING_BLOCKED 0x00000001	The fax service will not receive new incoming faxes.
FAX_OUTBOX_BLOCKED 0x00000002	The fax service will reject submissions of new outgoing faxes to its queue.
FAX_OUTBOX_PAUSED 0x00000004	The fax service will not dequeue and execute outgoing fax jobs from its queue.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x000003F7 ERROR_REGISTRY_CORRUPT	The registry is corrupted. The structure of one of the files containing registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.
0x80010101	Some required resource (for example, memory or events) could

Return value/code	Description
RPC_E_SYS_CALL_FAILED	not be allocated.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.91 FAX_SetReceiptsConfiguration (Opnum 35)

The **FAX_SetReceiptsConfiguration (Opnum 35)** method is called by the client. In response, the server sets the receipts configuration information that is used by the fax server to send delivery receipts for fax transmissions.<25>

```
error_status_t FAX_SetReceiptsConfiguration(
  [in] handle_t hFaxHandle,
  [in, ref] const PFAX_RECEIPTS_CONFIGW pReceipts
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

pReceipts: A pointer to a **FAX_RECEIPTS_CONFIGW (section 2.2.35)** object.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x000003F7 ERROR_REGISTRY_CORRUPT	The registry is corrupted. The structure of one of the files containing registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.
0x00001B63 FAX_ERR_NOT_SUPPORTED_ON_THIS_SKU	The fax server API version does not support the requested operation.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.92 FAX_SetRecipientsLimit (Opnum 83)

The **FAX_SetRecipientsLimit (Opnum 83)** method is called by the client. A fax client application calls the **FAX_SetRecipientsLimit** method to set the recipients limit of a single broadcast job. In response, the server sets the recipients limit of a single broadcast job.<26>

```
error_status_t FAX_SetRecipientsLimit(
  [in] handle_t hBinding,
  [in] DWORD dwRecipientsLimit
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwRecipientsLimit: A **DWORD** that specifies the maximum number of recipients for the fax.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000032 ERROR_NOT_SUPPORTED	Request is not supported.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.93 FAX_SetRoutingInfo (Opnum 16)

The **FAX_SetRoutingInfo (Opnum 16)** method is called by the client to set routing information for a fax routing method. In response, the server modifies the routing information for a fax routing method that is associated with a specific fax device.

The server MUST validate that the client has access to manage configuration on the server. The server MUST validate that the FaxHandle is not NULL. The server MUST validate that the RoutingGuid is for a valid routing method. The routing information MUST be passed on to the corresponding routing extension as specified by the RoutingGuid.

```
error_status_t FAX_SetRoutingInfo(
  [in] RPC_FAX_PORT_HANDLE FaxPortHandle,
  [in, string, unique] LPCWSTR RoutingGuid,
  [in, unique, size_is(RoutingInfoBufferSize)]
    const BYTE* RoutingInfoBuffer,
  [in, range (0,FAX_MAX_RPC_BUFFER)]
    DWORD RoutingInfoBufferSize
);
```

FaxPortHandle: An RPC context handle that references a specified fax port.

RoutingGuid: A pointer to a constant null-terminated Unicode string that specifies the globally unique identifier (GUID) that uniquely identifies the fax routing method to set the routing information for. This string must be in the form "{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}". Fax routing methods are defined by a fax routing extension, and the method is identified by a GUID. For more information about routing methods, see [MSDN-FRM].

The Microsoft fax routing provider defines the following routing methods.

Value	Meaning
REGVAL_RM_EMAIL_GUID "{6bbf7bfe-9af2-11d0-abf7-00c04fd91a4e}"	Set the information for routing faxes to e-mail. The *RoutingInfoBuffer* parameter is a pointer to a null-terminated Unicode string that contains the e-mail address that faxes are routed to.
REGVAL_RM_FOLDER_GUID "{92041a90-9af2-11d0-abf7-00c04fd91a4e}"	Set the information for routing faxes to a folder. The *RoutingInfoBuffer* parameter is a pointer to a null-terminated Unicode string that contains the path of the folder that faxes are routed to.
REGVAL_RM_PRINTING_GUID "{aec1b37c-9af2-11d0-abf7-00c04fd91a4e}"	Set the information for routing faxes to a printer. The *RoutingInfoBuffer* parameter is a pointer to a null-terminated Unicode string that contains the path and name of the printer that faxes are routed to.
REGVAL_RM_FLAGS_GUID "{aacc65ec-0091-40d6-a6f3-a2ed6057e1fa}"	Set the routing flags. The *RoutingInfoBuffer* parameter is a **DWORD** that contains the routing flags. The Microsoft fax routing provider defines the following routing flags. If this value is zero, none of these flags are set. Otherwise, this can be a combination of one or more of the following values.

Value	Meaning
0x00000001	Route faxes to a printer.
0x00000002	Route faxes to a folder.
0x00000004	Route faxes to the inbox. This flag is not supported.
0x00000008	Route faxes to an e-mail address.

RoutingInfoBuffer: A pointer to a buffer that contains the new fax routing information. The format and contents of this buffer depend on the routing method identified by the *RoutingGuid* parameter.

RoutingInfoBufferSize: The size, in bytes, of the *RoutingInfoBuffer* buffer.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000001 ERROR_INVALID_FUNCTION	The function is incorrect.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000000D ERROR_INVALID_DATA	The data is invalid.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.94 FAX_SetSecurity (Opnum 24)

The **FAX_SetSecurity (Opnum 24)** method is called by the client. In response, the server sets the fax server's security descriptor. Only an administrator with permissions can configure the security of the fax server.<27>

```
error_status_t FAX_SetSecurity(
  [in] handle_t hBinding,
  [in] SECURITY_INFORMATION SecurityInformation,
  [in, unique, size_is(dwBufferSize)]
    const LPBYTE pSecurityDescriptor,
  [in, range (0,FAX_MAX_RPC_BUFFER)]
    DWORD dwBufferSize
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

SecurityInformation: Identifies the components that are included in the security descriptor. The value of this parameter is a bitwise OR combination of **SECURITY_INFORMATION** constant values.

pSecurityDescriptor: A pointer to a **SECURITY_DESCRIPTOR** structure, as specified in [MS-DTYP] section 2, to be set.

dwBufferSize: A variable to return the size, in bytes, of the security descriptor.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000	Method is successful.

Return value/code	Description
ERROR_SUCCESS	
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000000D ERROR_INVALID_DATA	The data is invalid.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x000003F7 ERROR_REGISTRY_CORRUPT	The registry is corrupted. The structure of one of the files containing registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.

The server MUST validate that the client has the following credentials to set security on the server:

Action	Authorization
To set security information on object owned by the client	The right to change the owner in the object's security descriptor.
To set group security information	The right to modify the DACL in the object's security descriptor.
To set system wide security information	The right to modify the SACL in the object's security descriptor.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.95 FAX_SetSecurityEx2 (Opnum 100)

The **FAX_SetSecurityEx2 (Opnum 100)** method is called by the client. In response, the server sets the fax server's security descriptor. Only an administrator with permissions can configure the security of the fax server.<28>

This interface is supported only on Windows Vista or later. For earlier versions of Windows use FAX_SetSecurity.<29>

```
error_status_t FAX_SetSecurityEx2(
  [in] handle_t hBinding,
  [in] SECURITY_INFORMATION SecurityInformation,
  [in, unique, size_is(dwBufferSize)]
    const LPBYTE pSecurityDescriptor,
  [in, range (0,FAX_MAX_RPC_BUFFER)]
    DWORD dwBufferSize
);
```

hBinding: The fax server handle that is returned by a call to the **Fax_ConnectFaxServer (section 3.1.4.10)** method. This is a context handle.

SecurityInformation: Defines the desired entries, which are indicated as a bitwise OR operation, in the security descriptor to return.

pSecurityDescriptor: A pointer to a **SECURITY_DESCRIPTOR** structure, as specified in section 2 of [MS-DTYP].

dwBufferSize: A value that indicates the size, in bytes, of the *pSecurityDescriptor* buffer.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below:

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000000D ERROR_INVALID_DATA	The data is invalid.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x000003F7 ERROR_REGISTRY_CORRUPT	The registry is corrupted. The structure of one of the files that contains registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.96 FAX_StartCopyMessageFromServer (Opnum 69)

The **FAX_StartCopyMessageFromServer (Opnum 69)** method starts a copy process of a message from the server's archive or queue to the caller.

In response, the server MUST validate the message ID and the folder ID. The server MUST also validate that the client has access to query jobs in the queue folder or query messages in the archive folder.

To indicate success, the corresponding file (as specified by the message ID is opened), and the handle to the file is duplicated and returned to the client.

```
error_status_t FAX_StartCopyMessageFromServer(
  [in] handle_t hFaxHandle,
  [in] DWORDLONG dwlMessageId,
  [in] FAX_ENUM_MESSAGE_FOLDER Folder,
  [out, ref] PRPC_FAX_COPY_HANDLE lpHandle
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

dwlMessageId: A **DWORDLONG** value that indicates the message identifier to copy to the client.

Folder: This MUST be an enumeration value that indicates the folder from which to copy the message. For more information, see **FAX_ENUM_MESSAGE_FOLDER (section 2.2.1)**.

lpHandle: An RPC context handle to the file being copied.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below:

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x00001B61 FAX_ERR_MESSAGE_NOT_FOUND	The fax server cannot complete the operation because the number of active fax devices that are allowed for this version of Windows was exceeded.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.97 FAX_StartCopyToServer (Opnum 68)

The client calls the **FAX_StartCopyToServer (Opnum 68)** method to start a copy of a file to the server queue for which the client has access to submit either normal or high priority faxes. The server MUST generate a unique file name and create a file with that name on the queue. The server MUST duplicate the file handle and return it to the client to indicate success.

```
error_status_t FAX_StartCopyToServer(
  [in] handle_t hFaxHandle,
  [in, string, ref] LPCWSTR lpcwstrFileExt,
  [in, out, string, ref] LPWSTR lpwstrServerFileName,
  [out, ref] PRPC_FAX_COPY_HANDLE lpHandle
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpcwstrFileExt: A string containing the extension of the file to create on the server. The only file name extensions that are supported by the server are ".tif" or ".cov". The item is filtered by the file name extension.

lpwstrServerFileName: A string containing the file name with the specified extension of the file created on the server.

lpHandle: A context handle to the copied file.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.98 FAX_StartMessagesEnum (Opnum 63)

The **FAX_StartMessagesEnum (Opnum 63)** method is called by the client. In response, the server starts enumerating messages in one of the archives.

```
error_status_t FAX_StartMessagesEnum(
  [in] handle_t hFaxHandle,
  [in] FAX_ENUM_MESSAGE_FOLDER Folder,
  [out, ref] PRPC_FAX_MSG_ENUM_HANDLE lpHandle
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

Folder: This MUST be a **FAX_ENUM_MESSAGE_FOLDER (section 2.2.1)** enumeration that indicates the type of the archive where the message resides. The FAX_MESSAGE_FOLDER_QUEUE value is invalid for this parameter.

lpHandle: A pointer to an enumeration handle return value.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057	The parameter is incorrect.

Return value/code	Description
ERROR_INVALID_PARAMETER	
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.
0x00000103 ERROR_NO_MORE_ITEMS	No more data is available.

A fax client application calls the FAX_StartMessagesEnum function to start enumerating the message in one of the archives.

The server MUST validate that the client has access to the server. If this enumeration is attempted for all accounts, the server MUST validate that the client can query archives for all accounts. If the Inbox folder is specified and REASSIGN is turned on in the server, the server MUST ensure that the client has access to manage the incoming folder. On success, the server MUST create an enumeration handle and pass the same on to the client so that the latter might use the same for enumerating the messages.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.99 FAX_StartMessagesEnumEx (Opnum 90)

The **FAX_StartMessagesEnumEx (Opnum 90)** method is called by the client. In response, the server starts enumerating messages in one of the archives.

```
error_status_t FAX_StartMessagesEnumEx(
  [in] handle_t hFaxHandle,
  [in] BOOL fAllAccounts,
  [in, string, unique] LPCWSTR lpcwstrAccountName,
  [in] FAX_ENUM_MESSAGE_FOLDER Folder,
  [in] DWORD level,
  [out, ref] PRPC_FAX_MSG_ENUM_HANDLE lpHandle
);
```

hFaxHandle: The fax server handle that is returned by a call to the **Fax_ConnectFaxServer (section 3.1.4.10)** method. This is a context handle.

fAllAccounts: A flag indicating whether the messages for all accounts are enumerated. If this parameter is nonzero, the messages for all accounts are enumerated; otherwise, *lpcwstrAccountName* indicates which account is enumerated.

lpcwstrAccountName: A pointer to a constant null-terminated character string that indicates which account to enumerate. If this value is set to NULL, the current account's jobs are enumerated. Cross-account enumeration is currently not supported.

Folder: A **FAX_ENUM_MESSAGE_FOLDER (section 2.2.1)** enumeration that indicates the type of archive where the message resides. The FAX_MESSAGE_FOLDER_QUEUE value is invalid for this parameter.

level: A **DWORD** value that indicates the structure to return. This value must be set to 1.

lpHandle: A pointer to an enumeration handle return value.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.
0x0000006F ERROR_BUFFER_OVERFLOW	The file name is too long.
0x00000103 ERROR_NO_MORE_ITEMS	No more data is available.

A fax client application calls the FAX_StartMessagesEnumEx function to start enumerating messages in one of the archives. The enumerated messages have more information than those that are returned by FAX_StartMessagesEnum, namely, whether it has a cover page, the type of receipts selected, the e-mail address for receipts, and the flags from FAX_ENUM_MSG_FLAGS.

The account name that *lpcwstrAccountName* indicates must be in one of the following formats. Any other format is invalid.

Format	Description
<machine_name>\<user_name>	For a local user that has machine_name as the local machine's name.
<domain_name>\<user_name>	For a nonlocal user.

The server MUST validate that the client has some access to the server. If this enumeration is attempted for all accounts, the server MUST validate that the client can query all archives. If the Inbox folder is specified and REASSIGN is turned on in the server, the server MUST ensure that the client has the ability to manage the server folder. On success, the server MUST create an enumeration handle and pass the same on to the client so that the latter might use the same for enumerating the messages.<30>

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.100 FAX_StartServerNotification (Opnum 73)

The **FAX_StartServerNotification (Opnum 73)** method is called by the client. In response, the server should return 50 (ERROR_NOT_SUPPORTED).

```
error_status_t FAX_StartServerNotification(
  [in] handle_t hBinding,
  [in, string, ref] LPCWSTR lpcwstrMachineName,
  [in, string, ref] LPCWSTR lpcwstrEndPoint,
  [in] ULONG64 Context,
```

```
  [in, ref, string] LPCWSTR lpcwstrProtseqString,
  [in] BOOL bEventEx,
  [in] DWORD dwEventTypes,
  [out, ref] PRPC_FAX_EVENT_HANDLE lpHandle
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpcwstrMachineName: A pointer to a string that contains the name of the fax client machine. The machine name MUST be NULL for a local machine and a DNS name for a remote machine.

lpcwstrEndPoint: A pointer to a string that contains the client machine RPC server **endpoint** string. The endpoint MUST be a TCP port between 1024 and 65534 (in increments of 10).

Context: A **ULONG64** value that can be passed to **FAX_OpenConnection (section 3.2.4.4)** as a notification context.

lpcwstrProtseqString: A pointer to a string that contains the fax client RPC server's protocol sequence string. The protocol sequence string MUST be ncalrpc for local and ncan_ip_tcp for remote.

bEventEx: A Boolean value that indicates which notification method to use for notifications. This parameter is always set to **FALSE**.

dwEventTypes: A **DWORD** value that indicates which events the client wants to receive. This parameter is always set to 0. For more information, see **FAX_ENUM_EVENT_TYPE (section 2.2.49)**.

lpHandle: Returned notification context handle.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000032 ERROR_NOT_SUPPORTED	The request is not supported.

A fax client calls **FAX_StartServerNotification** (section 3.1.4.100) to inform the server that it wants to receive the notifications of fax events. The fax server should start an RPC client and call **FAX_OpenConnection** (section 3.2.4.4) on the client by using the supplied endpoint, protocol sequence information, and context handle information. The server then sends the notification of events to the client by using either **FAX_ClientEventQueueEx (section 3.2.4.2)** or **FAX_ClientEventQueue (section 3.2.4.1)**, as specified by the *bEventEx* parameter. When the client no longer wants to receive notifications, it calls **FAX_EndServerNotification (section 3.1.4.17)**, and the server should call **FAX_CloseConnection (section 3.2.4.3)** to close the connection.<31>

Note This method only supports TCP/IP as the transport protocol.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.101 FAX_StartServerNotificationEx (Opnum 74)

The **FAX_StartServerNotificationEx** method is called by the client. In response, the server starts the fax notification server. This method was added for Windows XP and later.

```
error_status_t FAX_StartServerNotificationEx(
  [in] handle_t hBinding,
  [in, string, ref] LPCWSTR lpcwstrMachineName,
  [in, string, ref] LPCWSTR lpcwstrEndPoint,
  [in] ULONG64 Context,
  [in, ref, string] LPCWSTR lpcwstrProtSeq,
  [in] BOOL bEventEx,
  [in] DWORD dwEventTypes,
  [out, ref] PRPC_FAX_EVENT_EX_HANDLE lpHandle
);
```

hBinding: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpcwstrMachineName: A pointer to a string containing the name of the fax client machine. The machine name MUST be NULL for a local machine and a DNS name for a remote machine.

lpcwstrEndPoint: A pointer to a string containing the client machine RPC server endpoint string. The endpoint MUST be a TCP port between 1024 and 65534 (in increments of 10).

Context: A **ULONG64** value that can be passed to **FAX_OpenConnection (section 3.2.4.4)** as a notification context.

lpcwstrProtSeq: A pointer to a string containing the fax client RPC server's protocol sequence string. On a Windows-based server, this parameter is ignored. The protocol used for sending the notifications is always TCP/IP. The protocol sequence string MUST be ncalrpc for local and ncan_ip_tcp for remote.

bEventEx: A **BOOLEAN** value that indicates which notification method to use for notifications. If set to **TRUE** the registration is for extended events (**FAX_EVENT_EX (section 2.2.53)** or **FAX_EVENT_EX_1 (section 2.2.54)**). If **FALSE**, the registration is for legacy events (**FAX_EVENT (section 2.2.52)**).

dwEventTypes: A **DWORD** value that indicates which events the client wants to receive. For more information, see **FAX_ENUM_EVENT_TYPE (section 2.2.49)**.

lpHandle: Returned extended notification context handle.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.

Return value/code	Description
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000000B ERROR_BAD_FORMAT	An attempt was made to load a program with an incorrect format.
0x0000000E ERROR_OUTOFMEMORY	Not enough storage is available to complete this operation.
0x0000001F ERROR_GEN_FAILURE	A device attached to the system is not functioning.

A fax client calls **FAX_StartServerNotificationEx** (section 3.1.4.101) to inform the server that it wants to receive the notifications of fax events. The fax server should start an RPC client and call **FAX_OpenConnection** (section 3.2.4.4) on the client by using the supplied endpoint, protocol sequence information, and context handle information. The server then sends notification of events to the client by using either **FAX_ClientEventQueueEx (section 3.2.4.2)** or **FAX_ClientEventQueue (section 3.2.4.1)** as specified by the *bEventEx* parameter. When the client no longer wants to receive notifications, it calls **FAX_EndServerNotification (section 3.1.4.17)**; the server should call **FAX_CloseConnection (section 3.2.4.3)** to close the connection.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.102 FAX_StartServerNotificationEx2 (Opnum 92)

The **FAX_StartServerNotificationEx2 (Opnum 92)** method is called by the client. In response, the server starts the fax notification server.

```
error_status_t FAX_StartServerNotificationEx2(
  [in] handle_t hBinding,
  [in, string, unique] LPCWSTR lpcwstrAccountName,
  [in, string, ref] LPCWSTR lpcwstrMachineName,
  [in, string, ref] LPCWSTR lpcwstrEndPoint,
  [in] ULONG64 Context,
  [in, ref, string] LPCWSTR lpcwstrProtseqString,
  [in] DWORD dwEventTypes,
  [in] DWORD level,
  [out, ref] PRPC_FAX_EVENT_EX_HANDLE lpHandle
);
```

hBinding: A fax server handle that is returned by a call to the **Fax_ConnectFaxServer (section 3.1.4.10)** method. This is a context handle.

lpcwstrAccountName: A pointer to a constant null-terminated character string that indicates which account to enumerate. If this value is NULL, the current account's jobs are enumerated. Cross-account enumeration is currently not supported.

lpcwstrMachineName: A pointer to a null-terminated string that contains the name of the fax client machine.

lpcwstrEndPoint: A pointer to a null-terminated string that contains the client machine remote procedure call (RPC) server endpoint string.

Context: A **ULONG64** value that can be passed to **FAX_OpenConnection (section 3.2.4.4)** as a notification context.

lpcwstrProtseqString: A pointer to a null-terminated string that contains the fax client RPC server's protocol sequence string. On a Windows-based server, this parameter is ignored. The protocol that is used for sending the notifications is always TCP/IP.

dwEventTypes: A **DWORD** value that indicates which events the client needs to receive. For more information, see **FAX_ENUM_EVENT_TYPE (section 2.2.49)**.

level: A **DWORD** value that indicates the structure to return. The value must be set to 1.

lpHandle: A pointer to an **RPC_FAX_EVENT_EX_HANDLE** (section 2.2.60) that returns an extended notification context handle.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x0000000B ERROR_BAD_FORMAT	An attempt was made to load a program with an incorrect format.
0x0000000E ERROR_OUTOFMEMORY	Not enough storage is available to complete this operation.
0x0000001F ERROR_GEN_FAILURE	A device that is attached to the system is not functioning.

The account name is the one on which to listen for events and a level that specifies the type of the structure that describes each event. The name lpcwstrAccountName is accessed only for account-based events.

The account name that *lpcwstrAccountName* indicates must be in one of the following formats. Any other format is invalid.

Format	Description
<machine_name>\<user_name>	For a local user that has machine_name as the name of the local machine.
<domain_name>\<user_name>	For a nonlocal user.

A fax client calls **FAX_StartServerNotificationEx2** (section 3.1.4.102) to inform the server that it needs to receive notifications of fax events. The fax server should start an RPC client and call **FAX_OpenConnection** (section 3.2.4.4) on the client by using the supplied endpoint, protocol sequence information, and context handle information. The server then sends notification of events

to the client by using either **FAX_ClientEventQueueEx (section 3.2.4.2)** or **FAX_ClientEventQueue (section 3.2.4.1)**, as specified by the *bEventEx* parameter. When the client no longer needs to receive notifications, it calls **FAX_EndServerNotification (section 3.1.4.17)**, and the server should call **FAX_CloseConnection (section 3.2.4.3)** to close the connection.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.103 FAX_UnregisterRoutingExtension (Opnum 62)

The **FAX_UnregisterRoutingExtension (Opnum 62)** method unregisters an existing inbound routing extension. For unregistration to take place, the fax server MUST be restarted.

```
error_status_t FAX_UnregisterRoutingExtension(
  [in] handle_t hFaxHandle,
  [in, string, ref] LPCWSTR lpctstrExtensionName
);
```

hFaxHandle: Specifies a fax server handle returned by a call to the **Fax_ConnectFaxServer (section 3.1.4.10)** method.

lpctstrExtensionName: Specifies the internal name of the fax routing extension.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF].

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.104 FAX_UnregisterServiceProviderEx (Opnum 61)

The **FAX_UnregisterServiceProviderEx (Opnum 61)** method is called when the client wants to unregister a fax service provider (FSP). In response, the server MUST validate that the client has write access. On success, the server MUST remove the service provider for the fax server.<32>

```
error_status_t FAX_UnregisterServiceProviderEx(
  [in] handle_t hFaxHandle,
  [in, string, ref] LPCWSTR lpctstrGUID
);
```

hFaxHandle: The handle that is provided by the client remote procedure call (RPC) layer when the RPC call is made.

lpctstrGUID: A pointer to a constant null-terminated character string that contains a valid string representation of the globally unique identifier of the fax service provider.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x0000005 ERROR_ACCESS_DENIED	Access is denied.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.4.105 FAX_WriteFile (Opnum 70)

The FAX_WriteFile (Opnum 70) method is called by the client. In response, the server receives a file and places it in the queue.

The server MUST validate the CopyHandle to be one that has been returned by a previous call to FAX_StartCopyToServer. The server MUST validate that the data size is not 0. On success, the server MUST copy the data to the server queue.

```
error_status_t FAX_WriteFile(
  [in, ref] RPC_FAX_COPY_HANDLE hCopy,
  [in, ref, size_is(dwDataSize)]
    const LPBYTE lpbData,
  [in, range(0,16384)] DWORD dwDataSize
);
```

hCopy: An RPC context handle that is returned by **FAX_StartCopyToServer (section 3.1.4.97)**.

lpbData: A pointer to the buffer from which to copy the file.

dwDataSize: A **DWORD** value indicating the size, in bytes, of the data buffer with a range of 0 to 16384.

Return Values: This method MUST return 0 (ERROR_SUCCESS) for success; otherwise, it MUST return one of the fax-specific errors that are defined in section 2.2.39 or one of the standard Windows errors that are defined in [MS-ERREF], the most common of which are listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000006 ERROR_INVALID_HANDLE	The handle is invalid.

Return value/code	Description
0x0000001F ERROR_GEN_FAILURE	A device attached to the system is not functioning.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.1.5 Timer Events

No protocol timer events are required on the server except the timers that are required in the underlying RPC protocol.

3.1.6 Other Local Events

This protocol does not attempt to reestablish a connection that is dropped by the lower layers.

3.2 Fax Client Details

3.2.1 Abstract Data Model

No abstract data model is required.

3.2.2 Timers

This protocol uses non-default behavior for the RPC Call Timeout timer that is defined in [MS-RPCE] section 3.3.2.2.2. The timer value that this protocol uses is 30000 milliseconds and it applies to all the methods that are described in this protocol.

3.2.3 Initialization

The server MUST listen on well-defined endpoints, as specified in [C706].

3.2.4 Message Processing Events and Sequencing Rules

The **Message Processing Events and Sequencing Rules** protocol MUST indicate to the RPC runtime that it is to perform a strict NDR data consistency check at target level 5.0, as specified in section 3 of [MS-RPCE].

Methods in RPC Opnum Order

Method	Description
FAX_OpenConnection	The **FAX_OpenConnection** method returns the context handle supplied by the FAX_StartServerNotification family of calls. This is done to provide a security layer, by verifying that the notifications are coming from an expected source Opnum: 0
FAX_ClientEventQueue	The fax client (the fax server acting as a client) calls this function when the

Method	Description
	client wants to deliver a fax event to this server. Opnum: 1
FAX_CloseConnection	The client (here the fax server acting as a client) calls this function when the client wants to release the connection to the server. When the server calls a FAX_EndServerNotification (section 3.1.4.17), the client MUST release the RPC connection to the server through this call. Opnum: 2
FAX_ClientEventQueueEx	The fax client (the fax server acting as a client) calls this function when the client wants to deliver a fax event to this server. Opnum: 3

All methods MUST NOT throw exceptions.

3.2.4.1 FAX_ClientEventQueue (Opnum 1)

The fax client (the fax server acting as a client) calls this function when the client wants to deliver a fax event to this server. The server registers for notifications with the client by calling one of FAX_StartServerNotification family of functions. In this call the server MUST pass an ASYNC_EVENT_INFO structure, which holds the signature of the server, and which the client MUST pass back to the server when it sends an event. This is done to provide a security layer, by verifying that the notifications are coming from an expected source.

In response, the server (the fax server is acting here as a client) MUST validate the signature in the FaxPortHandle argument that is sent by the client. If the validation fails, the server MUST abort the operation and MUST return ERROR_SUCCESS. If the signature is valid, the server MUST accept notifications for fax client events.

```
error_status_t FAX_ClientEventQueue(
  [in] RPC_FAX_HANDLE FaxPortHandle,
  [in] FAX_EVENT FaxEvent
);
```

FaxPortHandle: A fax Data Type indicating a context handle for this call.

FaxEvent: A FAX_EVENT (section 2.2.52) structure that contains the contents of an I/O completion packet. The fax server sends the completion packet to notify a fax client application about an asynchronous fax server event.

This method returns 0 (ERROR_SUCCESS) for success; otherwise, it returns one of the fax-specific errors or one of the return codes from Winerror.h, the most common of which is listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.2.4.2 FAX_ClientEventQueueEx (Opnum 3)

The fax client (the fax server is acting here as a client) calls this function when the client wants to deliver a fax event to this server. The server registers for notifications with the client by calling one of the FAX_StartServerNotification family of functions. In this call, the server MUST pass an ASYNC_EVENT_INFO structure, which holds the signature of the server, and which the client MUST pass back to the server when it sends an event. This is done to provide a security layer, by verifying that the notifications are coming from an expected source.

Data in **FAX_ClientEventQueueEx (Opnum 93)** (section 3.2.4.2) is serialized. Pointers to variable size data (such as strings) are replaced with offsets from the beginning of the buffer.

In response, the server (the fax server is acting here as a client) MUST validate the signature in the hClientContext argument, which is sent by the client. If the validation fails, the server MUST abort the operation and MUST return ERROR_SUCCESS. If the signature is valid, the server MUST accept notifications for fax client events.

```
error_status_t FAX_ClientEventQueueEx(
  [in, ref] RPC_FAX_HANDLE hClientContext,
  [in, ref, size_is(dwDataSize)]
    unsigned const char* lpbData,
  [in] DWORD dwDataSize
);
```

hClientContext: A fax Data Type indicating a context handle for this call.

lpbData: A pointer to a **FAX_EVENT_EX (section 2.2.53)** or **FAX_EVENT_EX_1 (section 2.2.54)** structure. The data is serialized. Pointers to variable size data (such as strings) are replaced with offsets from the beginning of the buffer.

dwDataSize: A DWORD containing the size of the buffer pointed to by the lpbData parameter.

This method returns 0 (ERROR_SUCCESS) for success; otherwise, it returns one of the fax-specific errors or one of the return codes from Winerror.h, the most common of which is listed below.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x0000000D ERROR_INVALID_DATA	The data is invalid.
0x0000000E ERROR_OUTOFMEMORY	Not enough storage is available to complete this operation.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

Data in FAX_ClientEventQueueEx is serialized. Pointers to variable size data (such as strings) are replaced with offsets from the beginning of the buffer.

3.2.4.3 FAX_CloseConnection (Opnum 2)

The client (the fax server is acting here as a client) calls this function when the client wants to release the connection to the server. When the server calls **FAX_EndServerNotification (Opnum 75) (section 3.1.4.17)**, the client MUST release the RPC connection to the server through this call.

In response, the server (the fax server is acting here as a client) MUST validate the signature in the FaxPortHandle argument that is sent by the client. If validation fails, the server MUST abort the operation, and MUST return ERROR_SUCCESS. If the signature is valid, the server MUST close the RPC connection that is identified by the argument.

```
error_status_t FAX_CloseConnection(
  [in, out] PRPC_FAX_HANDLE FaxHandle
);
```

FaxHandle: A pointer to an RPC_FAX_HANDLE that indicates a context handle to close. For more information about RPC_FAX_HANDLE, see fax Data Types.

This method returns 0 (ERROR_SUCCESS) for success; otherwise, it returns one of the fax-specific errors or one of the return codes from Winerror.h. No specialized error codes are associated with this method.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.

Exceptions Thrown:

No exceptions are thrown except those that are thrown by the underlying RPC protocol, [MS-RPCE].

3.2.4.4 FAX_OpenConnection (Opnum 0)

The **FAX_OpenConnection (Opnum 0)** method returns the context handle that is supplied by the FAX_StartServerNotification family of calls. This is done to provide a security layer, by verifying that the notifications are coming from an expected source.

In response, the server (in this case, a fax client acting in a server role) MUST validate the signature in the referred by the Context argument. The server MUST check for consistency in the data referred by *Context*. This is done to provide a security layer, by verifying that the notifications are coming from an expected source. The server MUST validate if the client uses an authentication scheme better than RPC_C_AUTHN_LEVEL_PKT_PRIVACY. An RPC client always uses a packet authentication level, as specified in [MS-RPCE] section 3.3.1.5.2.

On success, the server (in this case, a fax client acting in a server role) MUST open a notification session to the client and MUST return the same *Context* in the FaxHandle argument that was passed by the client in the FaxHandle argument.

```
error_status_t FAX_OpenConnection(
  [in] handle_t hBinding,
  [in] unsigned __int64 Context,
```

```
  [out] PRPC_FAX_HANDLE FaxHandle
);
```

hBinding: Handle provided by the client Remote Procedure Call (RPC) layer when the RPC call is made.

Context: A ULONG64 containing a context information handle. This handle should match the one supplied to the server when using the FAX_StartServerNotification family of calls. For more information, see the following topics.

- FAX_StartServerNotification (section 3.1.4.100)
- FAX_StartServerNotificationEx (section 3.1.4.101)
- FAX_StartServerNotificationEx2 (section 3.1.4.102)

FaxHandle: A pointer to an RPC_FAX_HANDLE indicating a context handle to open. This value is used in other fax client calls.

This method returns 0 (ERROR_SUCCESS) for success; otherwise, it returns one of the fax-specific errors or one of the return codes from Winerror.h.

Return value/code	Description
0x00000000 ERROR_SUCCESS	Method is successful.
0x00000005 ERROR_ACCESS_DENIED	The method requires at least packet-level privacy. The server checks the authentication level of the client. If it is less than RPC_C_AUTHN_LEVEL_PKT_PRIVACY, refuse access. Or there are other access-related problems.
0x00000057 ERROR_INVALID_PARAMETER	An invalid AssyncInfo structure is pointed to by the *Context* parameter or there are parameter-related problems.

Exceptions Thrown:

No exceptions are thrown except those thrown by the underlying RPC protocol, [MS-RPCE].

The returned PRPC_FAX_HANDLE is the *Context* parameter cast to a HANDLE.

The FAX_OpenConnection method returns the context handle supplied by the FAX_StartServerNotification family of calls. This is done to provide a security layer, by verifying that the notifications are coming from an expected source.

3.2.5 Timer Events

Not applicable.

3.2.6 Other Local Events

This protocol does not attempt to re-establish a connection if dropped by the lower layers.

4 Protocol Examples

The following section describes one or more operations as used in common scenarios to illustrate the function of the protocol.

4.1 Message Exchanges During Sending a Fax

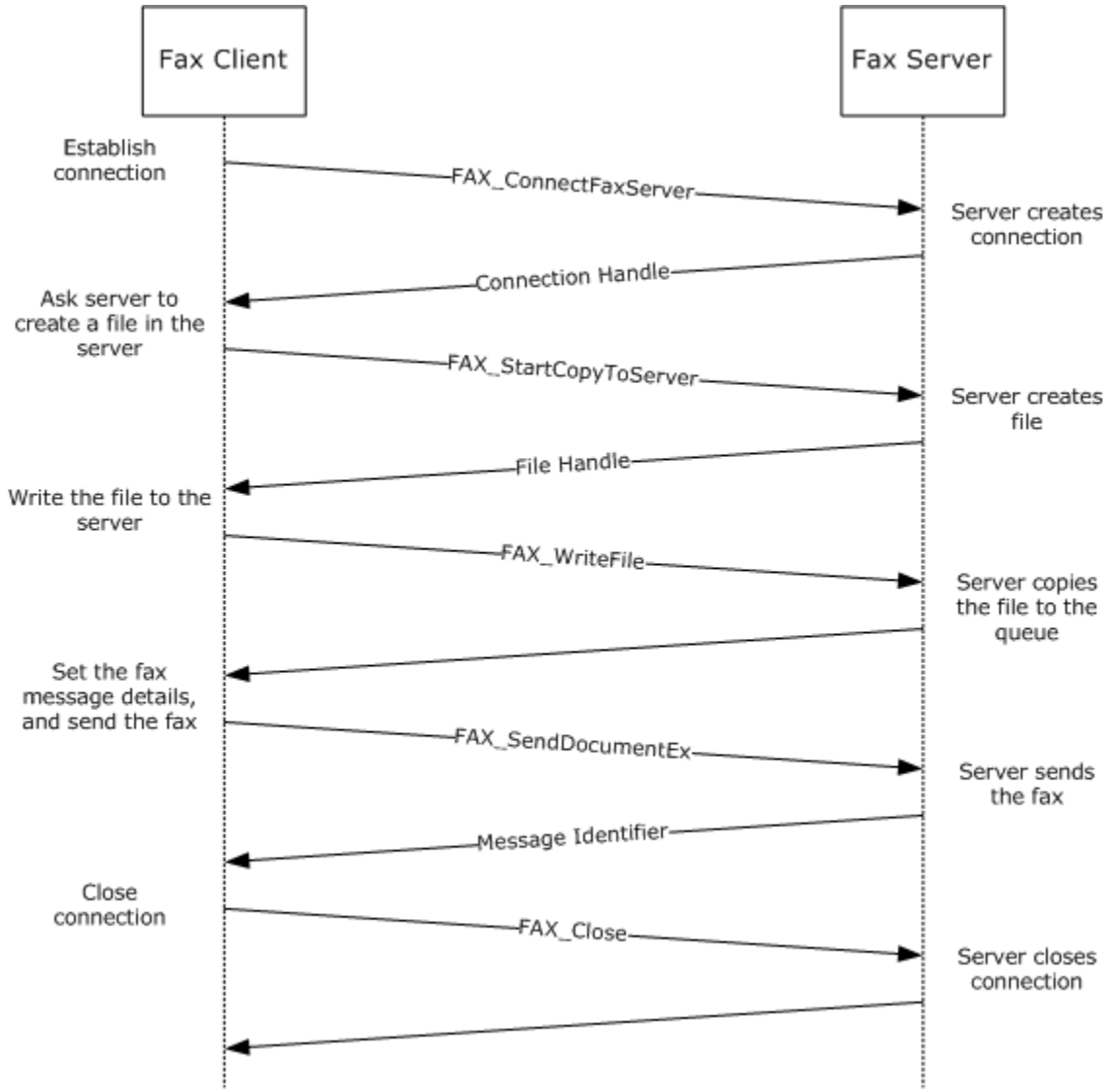

Figure 1: Message exchanges during the sending of a fax

A fax client follows these steps to send a fax using the fax server:

1. The client calls **FAX_ConnectFaxServer (section 3.1.4.10)** to establish a connection to the fax server. The parameters supplied to this function are the server name and a fax connection handle object. The server tries to establish the connection and returns **false** if the call fails, or sets the fax connection handle object if the call is successful.

The client does some of the following steps to create the fax message that needs to be sent:

- Setting the recipient information: To describe the recipients of the fax message, the client creates an array of **FAX_PERSONAL_PROFILE (section 2.2.33)**. The number of elements in the array is the number of recipients for the fax message.

  This structure is filled with the name of the recipient and the fax number.

- Setting the parameters of the fax transmission job: To set the parameters of the fax message transmission, the client can set the transmission-related fields, such as priority and receipt information, by using the **FAX_JOB_PARAM_EX (section 2.2.10)** structure.

- Setting the sender information: The sender's information that would be used with the fax message can be set by using the **FAX_PERSONAL_PROFILE** (section 2.2.33) structure.

- Setting the cover page: The fax client can set the cover page that would be used with the fax message by using the **FAX_COVERPAGE_INFO_EX (section 2.2.9)** structure.

- Setting the body: The client sets the body of the fax message.

2. The client calls **FAX_StartCopyToServer (section 3.1.4.97)** to request the server to create a file. The server creates the file and returns the file handle.

3. The client then uses the file handle obtained in step 3 and writes the file using the method **FAX_WriteFile (section 3.1.4.105)**. The server writes the file to the queue.

4. The client calls the **FAX_SendDocumentEx (section 3.1.4.73)** method to send the fax.

   The server tries to queue the fax for sending and returns **false** if the call fails. If the call is successful, the method returns **true** and sets the message identifier. The client can use the message identifier to track the status of the submitted fax message and control the fax transmission.

5. To end the connection to the fax server, the client calls FAX_Close by using the faxHandle parameter that was obtained in step 1.

4.2 Message Exchanges During Querying Server Configuration

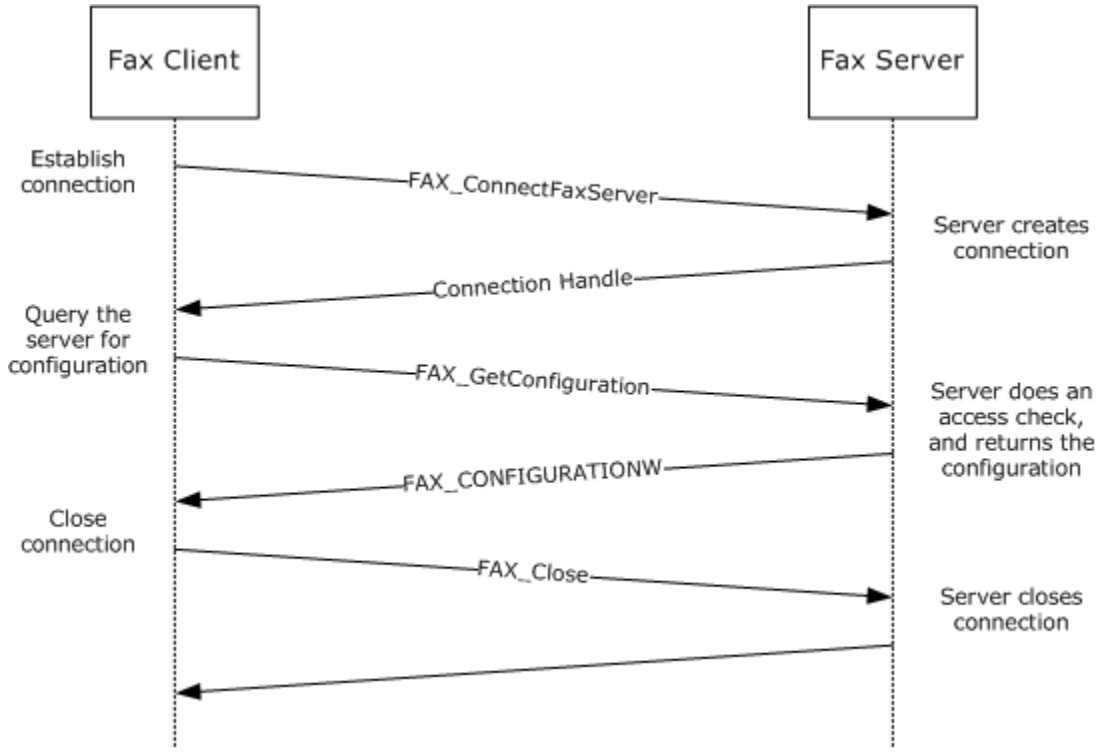

Figure 2: Message exchanges during the querying of server configuration

A fax client can query the server to obtain its global configuration. The client would do this to determine the global settings of the fax server. For example, the client can query for the number of retries, retry delays, and dirty days, and also for the branding and discount rate settings that are available as part of the **FAX_CONFIGURATION (section 2.2.21)** structure.

The client needs to have query configuration access to be able to query for the configuration settings on the server.

The client follows these steps to query for the global configuration settings on the fax server:

1. The client calls **FAX_ConnectFaxServer (section 3.1.4.10)** to establish a connection to the fax server. The parameters supplied to this function are the server name and a fax connection handle object. The server tries to establish the connection and returns false if the call fails or sets the fax connection handle object if successful.

2. The client calls **FAX_GetConfiguration (section 3.1.4.36)** to query the fax server configuration.

3. The server does an access check to determine whether the client has the permissions to query configuration. If the access check fails, the server returns ERROR_ACCESS_DENIED. If the client has the permissions to query for the server configuration and the call is successful, the server returns the **FAX_CONFIGURATION** (section 2.2.21) structure.

4. To end the connection to the fax server, the client calls FAX_Close by using the faxHandle parameter that was obtained in step 1.

4.3 Message Exchanges During Enumerating Fax Jobs

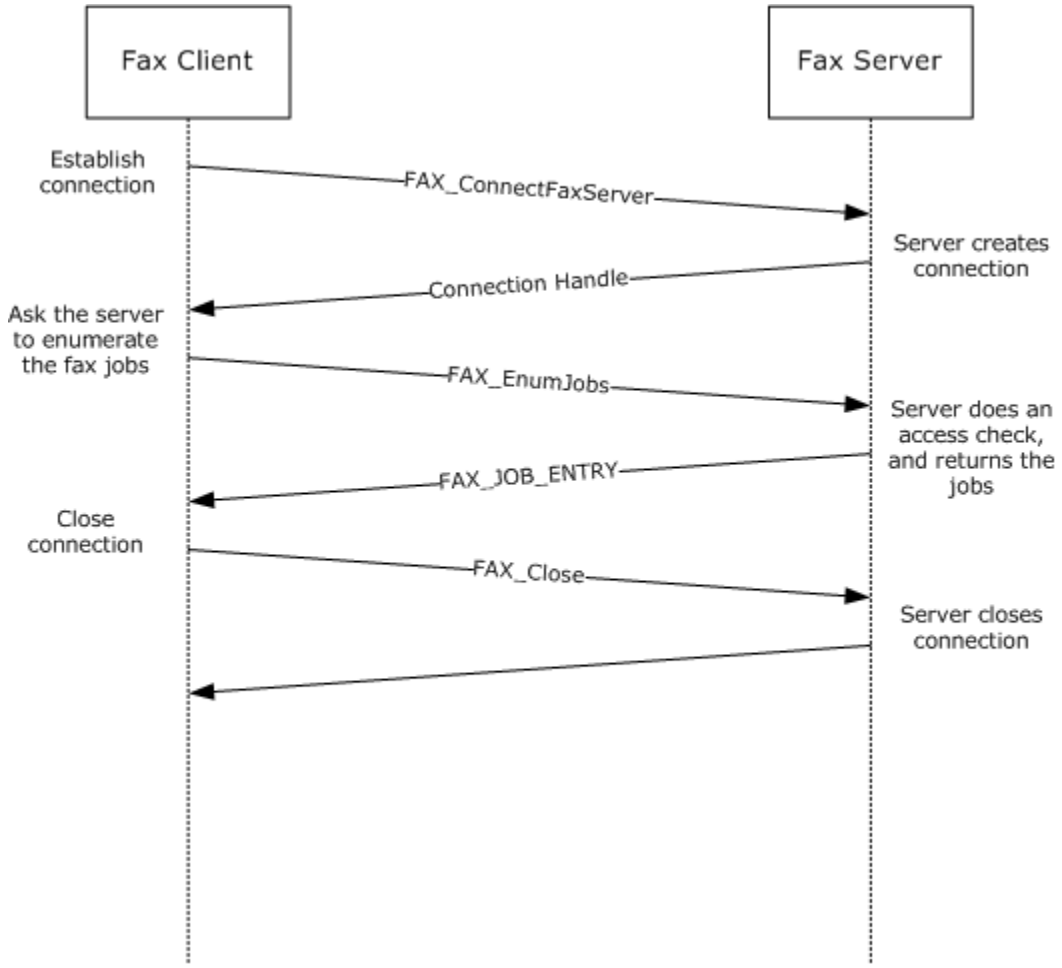

Figure 3: Message exchanges when enumerating fax jobs

A fax client can query the server to obtain a list of its queued and active fax jobs. To do so, the client follows these steps:

1. The client calls **FAX_ConnectFaxServer (section 3.1.4.10)** to establish a connection to the fax server. The parameters supplied to this function are the server name and a fax connection handle object. The server tries to establish the connection and returns false if the call fails or sets the fax connection handle object if successful.

2. The client calls **FAX_EnumJobs (section 3.1.4.21)** to query the list of fax jobs.

3. The server does an access check to determine whether the client has the permissions to enumerate server jobs. If the access check fails, the server returns ERROR_ACCESS_DENIED. If the client has the permissions to query for server configuration and the call is successful, the server returns the **FAX_JOB_ENTRY (section 2.2.4)** structure.

4. To end the connection to the fax server, the client calls FAX_Close by using the faxHandle parameter that was obtained in step 1.

When the client calls **FAX_EnumJobs** (section 3.1.4.21), it receives a list of jobs on the fax server queue. The client can access all the details of the jobs as defined in the **FAX_JOB_ENTRY** (section 2.2.4) structure. If the client already has the job ID of an inbound or an outbound job, the client can alternatively call the **FAX_GetJob (section 3.1.4.41)** function that returns the details of that particular job.

4.4 Message Exchanges During Modifying Fax Jobs

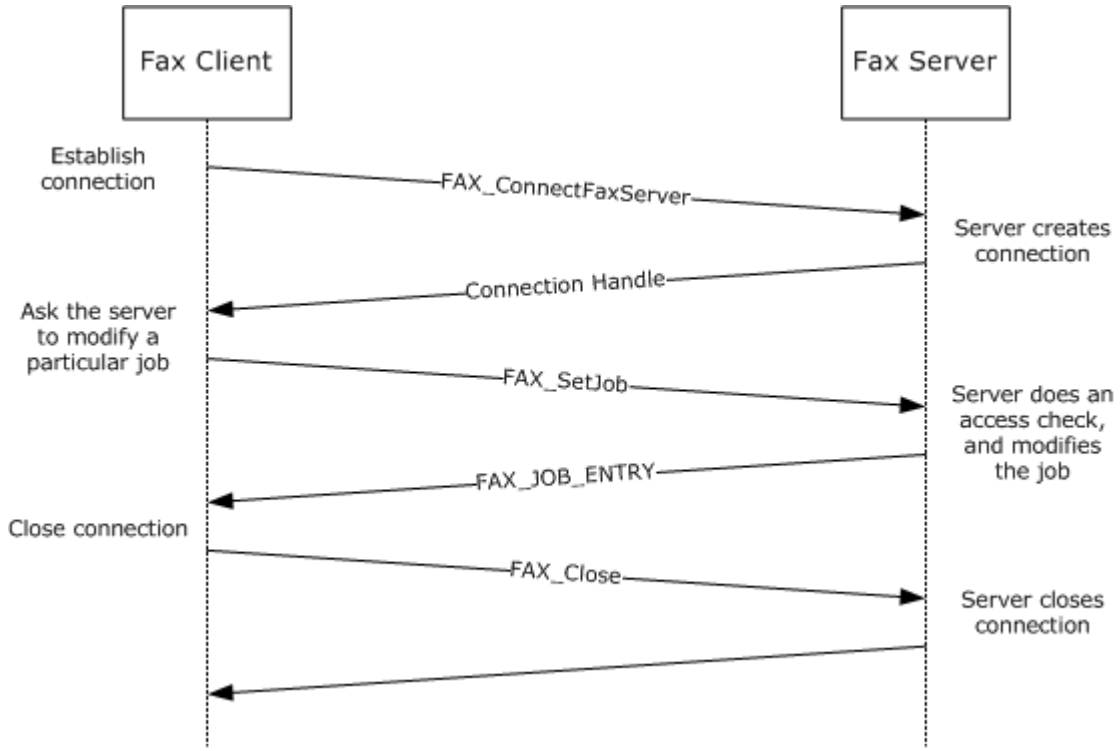

Figure 4: Message exchanges when modifying fax jobs

A fax client cannot modify the attributes or properties of a fax transmission after the job has been queued. However, the client can pause, resume, cancel, or restart a queued fax job. To do so, the client follows these steps:

1. The client calls **FAX_ConnectFaxServer (section 3.1.4.10)** to establish a connection to the fax server. The parameters supplied to this function are the server name and a fax connection handle object. The server tries to establish the connection and returns false if the call fails or sets the fax connection handle object if successful.

2. The client calls **FAX_SetJob (section 3.1.4.82)** by using the particular job Id that the client wants to modify. As part of the method, the client passes the command that it wants to execute: delete, pause, resume, or restart.

3. The server does an access check to determine whether the client has the permissions to modify server jobs. If the access check fails, the server returns ERROR_ACCESS_DENIED. If the client has the permissions to modify the job, the server does the modification and returns nonzero to indicate success or zero to indicate failure.

4. To end the connection to the fax server, the client calls FAX_Close by using the faxHandle parameter that was obtained in step 1.

4.5 Message Exchanges During Adding an Outbound Routing Rule

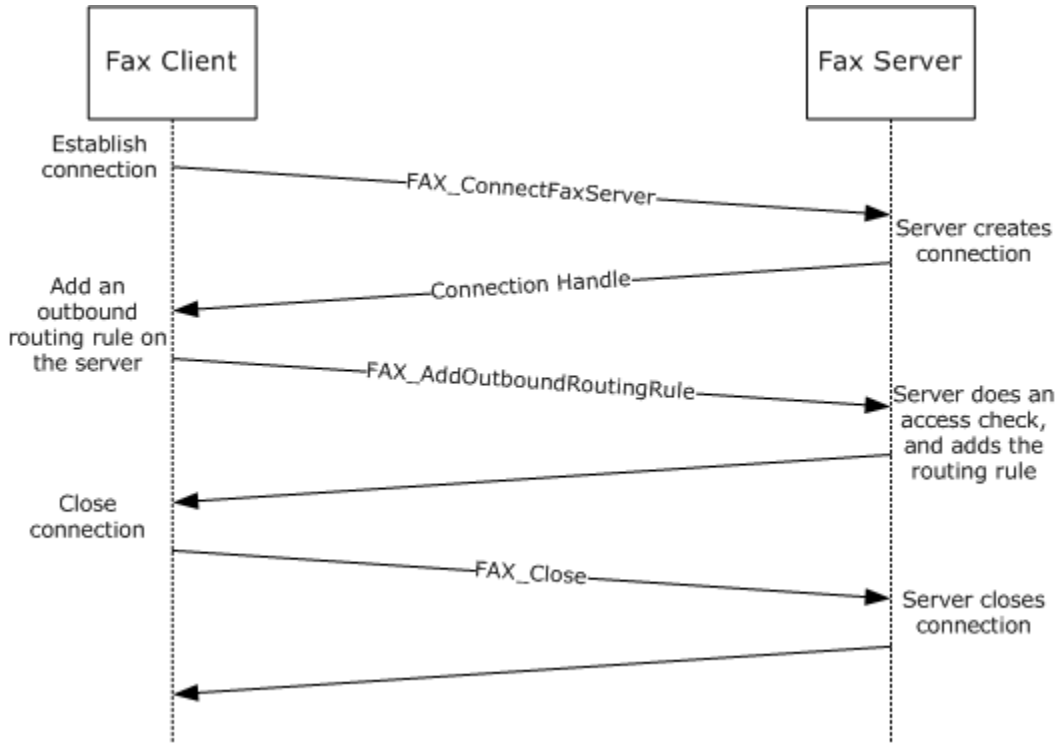

Figure 5: Message exchanges when adding an outbound routing rule

A fax client can add an outbound routing rule on the server. To do so, the fax client does the following:

1. The client calls **FAX_ConnectFaxServer (section 3.1.4.10)** to establish a connection to the fax server. The parameters supplied to this function are the server name and a fax connection handle object. The server tries to establish the connection and returns false if the call fails or sets the fax connection handle object if successful.

2. The client calls **FAX_AddOutboundRule (section 3.1.4.5)** to add an outbound routing rule on the server. The client passes the following parameters:

   - A handle to the fax connection.
   - The area code of the outbound routing rule.
   - The country/region code of the outbound routing rule.
   - The destination device ID of the rule.
   - The destination group of the rule.
   - A Boolean value that specifies whether the group should be used as the destination.

3. The server does an access check to determine whether the client has the permissions to add an outbound routing rule. If the access check fails, the server returns ERROR_ACCESS_DENIED. If the client has the permissions to add the rule, the server does the modification and returns a zero to indicate success.

4. To end the connection to the fax server, the client calls FAX_Close by using the faxHandle parameter that was obtained in step 1.

4.6 Message Exchanges During Registering and Unregistering for Server Notifications

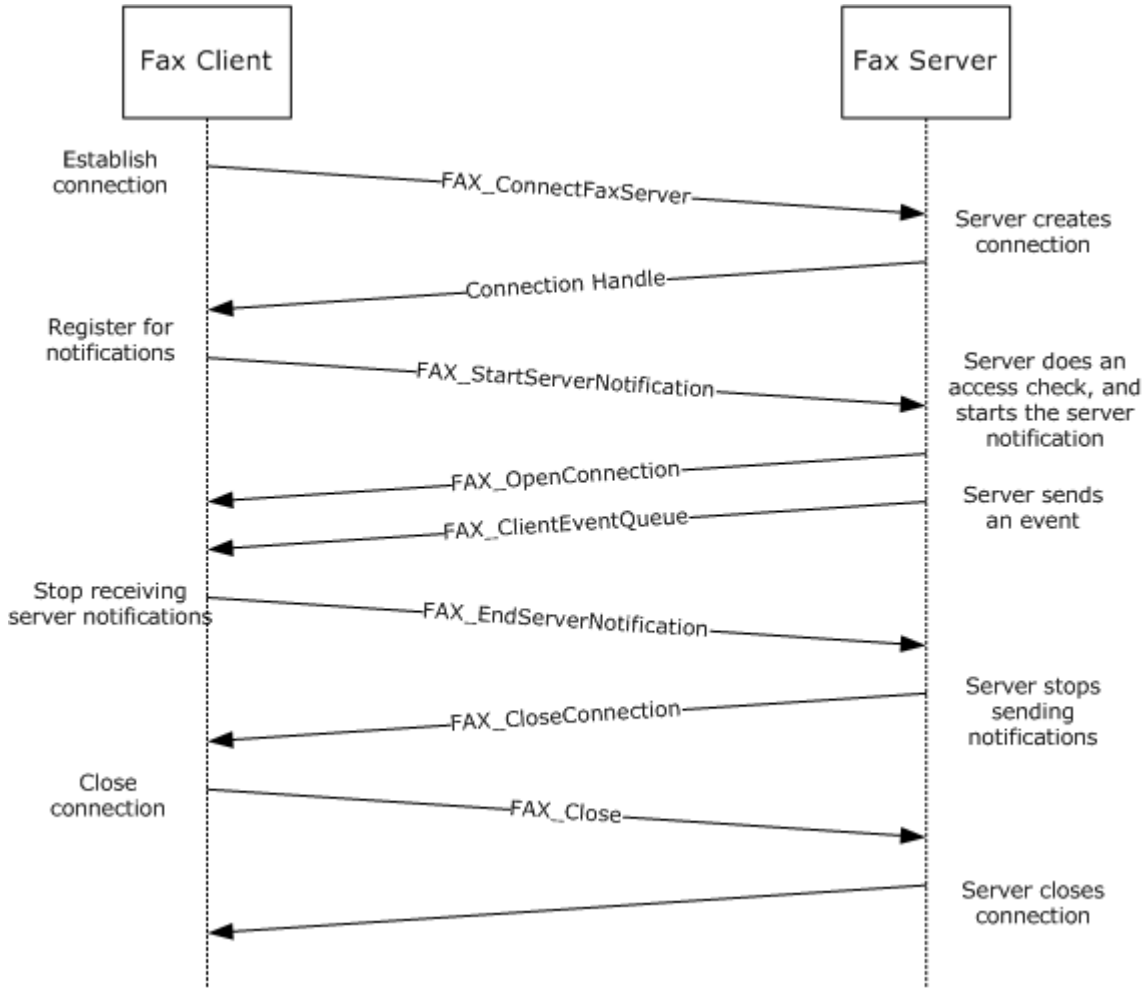

Figure 6: Message exchanges when registering and unregistering for server notifications

A fax client can inform the server that it wants to receive the notifications of fax events. To do so, the client follows these steps:

1. The client calls **FAX_ConnectFaxServer (section 3.1.4.10)** to establish a connection to the fax server. The parameters supplied to this function are the server name and a fax connection handle object. The server tries to establish the connection and returns **false** if the call fails or sets the fax connection handle object if successful.

2. The client calls the **FAX_StartServerNotification (section 3.1.4.100)** method to register for server notifications. The client passes the connection handle, the name of the fax client machine, a pointer to a string containing the client machine RPC server endpoint, and a pointer to a string that contains the fax client RPC server's protocol sequence string, among other parameters.

3. The fax server starts an RPC client and calls **Fax_OpenConnection (section 3.2.4.4)** by using the supplied endpoint, protocol sequence information, and context handle information.

4. The fax server sends a notification of events to the client by using the **Fax_ClientEventQueue (section 3.2.4.1)** method.

5. When the client no longer wants to receive notifications, it calls **FAX_EndServerNotification (section 3.1.4.17)**.

6. The server calls **Fax_CloseConnection (section 3.2.4.3)** to close the connection with the client.

7. To end the connection to the fax server, the client calls FaxClose by using the faxHandle parameter that was obtained in step 1.

4.7 Message Exchanges During Granting Security Privileges to a User

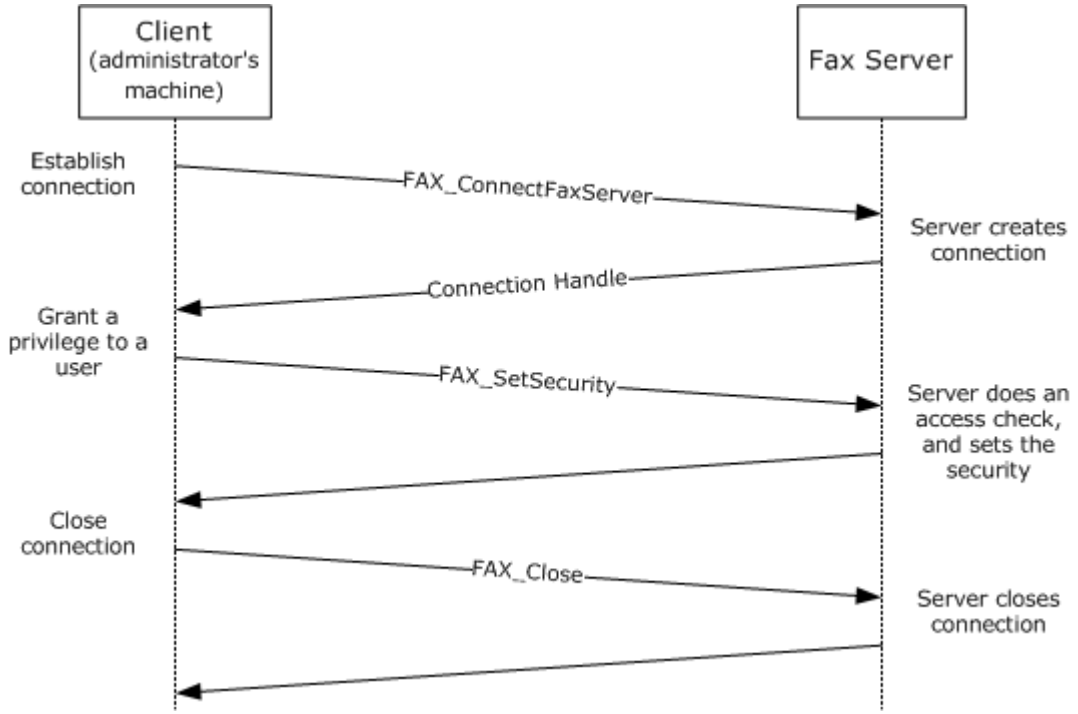

Figure 7: Message exchanges when granting security privileges to a user

A user requests an additional privilege from a fax administrator (for example, a user might request permission to send high-priority faxes). The fax administrator follows these steps:

1. From a client machine, the fax administrator calls **FAX_ConnectFaxServer (section 3.1.4.10)** to establish a connection to the fax server. The parameters supplied to this function are the server name and a fax connection handle object. The server tries to establish the connection and returns false if the call fails or sets the fax connection handle object if successful.

2. The client calls the **FAX_SetSecurity (section 3.1.4.94)** method, by passing the connection handle, the components that are included in the security descriptor, and a **SECURITY_DESCRIPTOR** structure that contains the security attributes to be set.

3. The server does an access check to determine whether the client that is calling the method has the access permissions to modify the security on the fax server.

4. The server sets the security as requested in the **SECURITY_DESCRIPTOR** and returns zero to indicate success.

5. To end the connection to the fax server, the client calls FAX_Close by using the faxHandle parameter that was obtained in step 1.

5 Security

The following sections specify security considerations for administrators.

5.1 Security Considerations for Implementers

Security considerations for both authenticated and unauthenticated RPC used in this protocol are as specified in [MS-RPCE]. The client SHOULD always perform authenticated RPC.

The RPC connection uses ncalrpc procotol in case of a Local Fax call and ncacn_ip_tcp in case of connection with a remote Fax server. The RPC connection is made using RPC_C_AUTHN_LEVEL_PKT_PRIVACY. The packet authentication level is as specified in [MS-RPCE] section 3.3.1.5.2.<33>

The server SHOULD perform access control checks based on the credentials of the user.<34>

5.2 Index of Security Parameters

This protocol defines no security parameters.

6 Appendix A: Full IDL

For ease of implementation, the full **IDLs** for all interfaces defined in this protocol are provided in this appendix.

6.1 Appendix A.1: FaxServer IDL

For ease of implementation, the full IDL is provided below, where "ms-dtyp.idl" is the IDL found in [MS-DTYP] Appendix A. The file "imports.idl" is the IDL defined in section 6.

```
import "ms-dtyp.idl";

interface fax_imports;
interface fax;

#define HCALL DWORD
#define  FAX RPC LIMIT H
#define FAX_MAX_RPC_BUFFER                (1024 * 1024)
#define FAX_MAX_DEVICES_IN_GROUP          1000
#define FAX MAX RECIPIENTS                 10000
#define RPC COPY BUFFER SIZE   16384

typedef [context handle] HANDLE RPC FAX PORT HANDLE;
typedef RPC_FAX_PORT_HANDLE *PRPC_FAX_PORT_HANDLE;

typedef [context handle] HANDLE RPC FAX SVC HANDLE;
typedef RPC FAX SVC HANDLE *PRPC FAX SVC HANDLE;

typedef [context handle] HANDLE RPC FAX MSG ENUM HANDLE;
typedef RPC_FAX_MSG_ENUM_HANDLE *PRPC_FAX_MSG_ENUM_HANDLE;

typedef [context handle] HANDLE RPC FAX COPY HANDLE;
typedef RPC FAX COPY HANDLE *PRPC FAX COPY HANDLE;

typedef [context_handle] HANDLE RPC_FAX_EVENT_HANDLE;
typedef RPC_FAX_EVENT_HANDLE *PRPC_FAX_EVENT_HANDLE;

typedef [context handle] HANDLE RPC FAX EVENT EX HANDLE;
typedef RPC FAX EVENT EX HANDLE *PRPC FAX EVENT EX HANDLE;

#ifdef SERVER_STUB
  midl_pragma warning( disable: 2466 2467 )
  typedef [range(0, RPC COPY BUFFER SIZE)]    DWORD    RANGED DWORD;
  typedef RANGED DWORD *    LPRANGED DWORD;
#else
  typedef DWORD *    LPRANGED DWORD;
#endif

typedef struct {
  DWORD dwSizeOfStruct;
  DWORD dwCoverPageFormat;
  LPWSTR lptstrCoverPageFileName;
  BOOL bServerBased;
  LPWSTR lptstrNote;
  LPWSTR lptstrSubject;
} FAX COVERPAGE INFO EXW,
  *PFAX COVERPAGE INFO EXW,
  *LPCFAX COVERPAGE INFO EXW;

typedef struct {
  DWORD dwSizeOfStruct;
  LPWSTR lptstrName;
  LPWSTR lptstrFaxNumber;
```

```
  LPWSTR lptstrCompany;
  LPWSTR lptstrStreetAddress;
  LPWSTR lptstrCity;
  LPWSTR lptstrState;
  LPWSTR lptstrZip;
  LPWSTR lptstrCountry;
  LPWSTR lptstrTitle;
  LPWSTR lptstrDepartment;
  LPWSTR lptstrOfficeLocation;
  LPWSTR lptstrHomePhone;
  LPWSTR lptstrOfficePhone;
  LPWSTR lptstrEmail;
  LPWSTR lptstrBillingCode;
  LPWSTR lptstrTSID;
} FAX_PERSONAL_PROFILEW,
  *LPCFAX PERSONAL PROFILEW;

typedef struct  FAX JOB PARAM {
    DWORD SizeOfStruct;
    LPCTSTR RecipientNumber;
    LPCTSTR RecipientName;
    LPCTSTR Tsid;
    LPCTSTR SenderName;
    LPCTSTR SenderCompany;
    LPCTSTR SenderDept;
    LPCTSTR BillingCode;
    DWORD ScheduleAction;
    SYSTEMTIME ScheduleTime;
    DWORD DeliveryReportType;
    LPCTSTR DeliveryReportAddress;
    LPCTSTR DocumentName;
    HCALL CallHandle;
    DWORD_PTR Reserved[3];
} FAX JOB PARAM, *PFAX JOB PARAM;

typedef enum
{
  FAX_DEVICE_RECEIVE_MODE_OFF = 0,
  FAX_DEVICE_RECEIVE_MODE_AUTO = 1,
  FAX DEVICE RECEIVE MODE MANUAL = 2
}FAX ENUM DEVICE RECEIVE MODE;

typedef enum
{
    FAX GROUP STATUS ALL DEV VALID,
    FAX GROUP STATUS EMPTY,
    FAX GROUP STATUS ALL DEV NOT VALID,
    FAX GROUP STATUS SOME DEV NOT VALID,
} FAX_ENUM_GROUP_STATUS;

typedef enum
{
    FAX MESSAGE FOLDER INBOX,
    FAX MESSAGE FOLDER SENTITEMS,
    FAX_MESSAGE_FOLDER_QUEUE
} FAX_ENUM_MESSAGE_FOLDER;

typedef enum
{
  RECIPIENT PERSONAL PROF = 1,
  SENDER PERSONAL PROF = 2
}FAX_ENUM_PERSONAL_PROF_TYPES;


typedef enum
```

```
{
    FAX_PRIORITY_TYPE_LOW,
    FAX_PRIORITY_TYPE_NORMAL,
    FAX_PRIORITY_TYPE_HIGH
} FAX_ENUM_PRIORITY_TYPE;

typedef enum
{
  FAX_SMTP_AUTH_ANONYMOUS = 0,
  FAX_SMTP_AUTH_BASIC = 1,
  FAX_SMTP_AUTH_NTLM = 2
}FAX_ENUM_SMTP_AUTH_OPTIONS;

typedef enum
{
  PRODUCT_SKU_UNKNOWN = 0x00000000,
  PRODUCT_SKU_PERSONAL = 0x00000001,
  PRODUCT_SKU_PROFESSIONAL = 0x00000002,
  PRODUCT_SKU_SERVER = 0x00000004,
  PRODUCT_SKU_ADVANCED_SERVER = 0x00000008,
  PRODUCT_SKU_DATA_CENTER = 0x00000010,
  PRODUCT_SKU_DESKTOP_EMBEDDED = 0x00000020,
  PRODUCT_SKU_SERVER_EMBEDDED = 0x00000040,
  PRODUCT_SKU_WEB_SERVER = 0x00000080
}PRODUCT_SKU_TYPE;

typedef enum
{
  FAX_CONFIG_OPTION_ALLOW_PERSONAL_CP = 0x00000000,
  FAX_CONFIG_OPTION_QUEUE_STATE = 0x00000001,
  FAX_CONFIG_OPTION_ALLOWED_RECEIPTS = 0x00000002,
  FAX_CONFIG_OPTION_INCOMING_FAXES_PUBLIC = 0x00000003
}FAX_ENUM_CONFIG_OPTION;

typedef struct {
  WORD Hour;
  WORD Minute;
} FAX_TIME,
 *PFAX_TIME;

typedef struct {
  DWORD dwSizeOfStruct;
  DWORD dwAllowedReceipts;
  FAX_ENUM_SMTP_AUTH_OPTIONS SMTPAuthOption;
  LPWSTR lptstrReserved;
  LPWSTR lptstrSMTPServer;
  DWORD dwSMTPPort;
  LPWSTR lptstrSMTPFrom;
  LPWSTR lptstrSMTPUserName;
  LPWSTR lptstrSMTPPassword;
  BOOL bIsToUseForMSRouteThroughEmailMethod;
} FAX_RECEIPTS_CONFIGW,
  *PFAX_RECEIPTS_CONFIGW;

typedef struct {
    DWORD SizeOfStruct;
    DWORD Retries;
    DWORD RetryDelay;
    BOOL Branding;
    DWORD DirtyDays;
    BOOL UseDeviceTsid;
    BOOL ServerCp;
    BOOL PauseServerQueue;
    FAX_TIME StartCheapTime;
    FAX_TIME StopCheapTime;
```

```
    BOOL ArchiveOutgoingFaxes;
    LPCTSTR ArchiveDirectory;
    LPCSTR InboundProfile;
} FAX CONFIGURATIONW,
  *PFAX CONFIGURATIONW;

typedef struct {
  DWORD SizeOfStruct;
  DWORD Priority;
  LPCWSTR Guid;
  LPCWSTR FriendlyName;
  LPCWSTR FunctionName;
  LPCWSTR ExtensionImageName;
  LPCWSTR ExtensionFriendlyName;
} FAX_GLOBAL_ROUTING_INFOW;

typedef struct  FAX JOB PARAM EXW
{
        DWORD                   dwSizeOfStruct;
        DWORD                   dwScheduleAction;
        SYSTEMTIME              tmSchedule;
        DWORD                   dwReceiptDeliveryType;
        LPWSTR                  lptstrReceiptDeliveryAddress;
        FAX ENUM PRIORITY TYPE  Priority;
        HCALL                   hCall;
        DWORD_PTR               dwReserved[4];
        LPWSTR                  lptstrDocumentName;
        DWORD                   dwPageCount;
} FAX JOB PARAM EXW,
  *PFAX JOB PARAM EXW,
  *LPCFAX JOB PARAM EXW;

typedef struct _RPC_FAX_OUTBOUND_ROUTING_GROUPW
{
    DWORD                                       dwSizeOfStruct;
    [string] LPWSTR                             lpwstrGroupName;
    [range(0,FAX MAX DEVICES IN GROUP)]DWORD    dwNumDevices;
    [unique, size_is(dwNumDevices)] LPDWORD     lpdwDevices;
    FAX_ENUM_GROUP_STATUS                       Status;
} RPC FAX OUTBOUND ROUTING GROUPW,
  *PRPC FAX OUTBOUND ROUTING GROUPW;


typedef struct _FAX_PORT_INFO
{
    DWORD                           dwSizeOfStruct;
    DWORD                           dwDeviceID;
    LPCWSTR                         lpctstrDeviceName;
    LPWSTR                          lptstrDescription;
    LPCWSTR                         lpctstrProviderName;
    LPCWSTR                         lpctstrProviderGUID;
    BOOL                            bSend;
    FAX ENUM DEVICE RECEIVE MODE    ReceiveMode;
    DWORD                           dwStatus;
    DWORD                           dwRings;
    LPWSTR                          lptstrCsid;
    LPWSTR                          lptstrTsid;
} FAX PORT INFO, *PFAX PORT INFO;

typedef [switch type(int)] union  FAX RULE DESTINATION
{
    [case(0)]
        DWORD               dwDeviceId;
    [default]
        [string] LPWSTR     lpwstrGroupName;
```

```
} FAX RULE DESTINATION;

typedef struct {
    DWORD                                       dwSizeOfStruct;
    DWORD                                       dwAreaCode;
    DWORD                                       dwCountryCode;
    [string] LPWSTR                             lpwstrCountryName;
    [switch_is(bUseGroup)] FAX_RULE_DESTINATION  Destination;
    BOOL                                        bUseGroup;
} RPC FAX OUTBOUND ROUTING RULEW,
  *PRPC FAX OUTBOUND ROUTING RULEW;

typedef struct {
  DWORD dwSizeOfStruct;
  BOOL bValid;
  WORD wMajorVersion;
  WORD wMinorVersion;
  WORD wMajorBuildNumber;
  WORD wMinorBuildNumber;
  DWORD dwFlags;
} FAX VERSION,
 *PFAX VERSION;

typedef struct {
  DWORD dwSizeOfStruct;
  BOOL bAllowPersonalCP;
  BOOL bUseDeviceTSID;
  DWORD dwRetries;
  DWORD dwRetryDelay;
  FAX TIME dtDiscountStart;
  FAX TIME dtDiscountEnd;
  DWORD dwAgeLimit;
  BOOL bBranding;
} FAX OUTBOX CONFIG,
 *PFAX OUTBOX CONFIG;

typedef struct {
  DWORD dwSizeOfStruct;
  BOOL bUseArchive;
  LPWSTR lpcstrFolder;
  BOOL bSizeQuotaWarning;
  DWORD dwSizeQuotaHighWatermark;
  DWORD dwSizeQuotaLowWatermark;
  DWORD dwAgeLimit;
  DWORDLONG dwlArchiveSize;
} FAX ARCHIVE CONFIGW,
 *PFAX ARCHIVE CONFIGW;

typedef struct {
  DWORD dwSizeOfStruct;
  BOOL bLogIncoming;
  BOOL bLogOutgoing;
  LPWSTR lptstrDBPath;
} FAX ACTIVITY LOGGING CONFIGW,
 *PFAX_ACTIVITY_LOGGING_CONFIGW;

typedef struct {
  DWORD dwSizeOfStruct;
  DWORD dwDeviceID;
  LPCWSTR lpctstrDeviceName;
  LPWSTR lptstrDescription;
  LPCWSTR lpctstrProviderName;
  LPCWSTR lpctstrProviderGUID;
  BOOL bSend;
  FAX_ENUM_DEVICE_RECEIVE_MODE ReceiveMode;
```

```
  DWORD dwStatus;
  DWORD dwRings;
  LPWSTR lptstrCsid;
  LPWSTR lptstrTsid;
} FAX PORT INFO EXW,
 *PFAX PORT INFO EXW;

typedef struct {
  DWORD dwSizeOfStruct;
  DWORD dwIncomingMessages;
  DWORD dwRoutingMessages;
  DWORD dwOutgoingMessages;
  DWORD dwDelegatedOutgoingMessages;
  DWORD dwQueuedMessages;
  DWORD dwErrorEvents;
  DWORD dwWarningEvents;
  DWORD dwInformationEvents;
} FAX SERVER ACTIVITY,
 *PFAX SERVER ACTIVITY;

typedef struct {
  LPCWSTR lpcwstrRecipients;
  LPCWSTR lpcwstrSenderName;
  LPCWSTR lpcwstrSenderFaxNumber;
  LPCWSTR lpcwstrSubject;
  BOOL bHasCoverPage;
} FAX_REASSIGN_INFO,
 *PFAX REASSIGN INFO;

typedef struct  {
  DWORD dwValidityMask;
  DWORD dwMsgFlags;
} FAX_MESSAGE_PROPS,
 *PFAX MESSAGE PROPS;


[
    local,
#ifdef __midl
        ms union,
#endif //   midl
 version(1.0)
]
interface fax_imports

{
#define MIDL PASS
DWORD
Dummy(
    [in]    DWORD   DummyParm);
}

[
    uuid(ea0a3165-4834-11d2-a6f8-00c04fa346cc),
    version(4.0),
    pointer_default(unique)
]

interface fax

{

error_status_t
FAX GetServicePrinters(
    [in] handle_t hBinding,
```

```
    [out, size is(,*lpdwBufferSize)] LPBYTE *lpBuffer,
    [out, ref] LPDWORD lpdwBufferSize,
    [out, ref] LPDWORD lpdwPrintersReturned
    );


error status t
FAX_ConnectionRefCount(
    [in] handle_t hBinding,
    [in, out] PRPC FAX SVC HANDLE Handle,
    [in] DWORD Connect,
    [out] LPDWORD CanShare
    );

error_status_t
FAX OpenPort(
   [in] handle t hBinding,
   [in] DWORD DeviceId,
   [in] DWORD Flags,
   [out] PRPC_FAX_PORT_HANDLE FaxPortHandle
   );

error status t
FAX ClosePort(
   [in,out] PRPC_FAX_PORT_HANDLE FaxPortHandle
   );

error status t
FAX EnumJobs(
   [in] handle t hBinding,
   [out, size is(,*BufferSize)] LPBYTE *Buffer,
   [out, ref] LPDWORD BufferSize,
   [out, ref] LPDWORD JobsReturned
   );

error status t
FAX GetJob(
   [in] handle_t hBinding,
   [in] DWORD JobId,
   [out, size is(,*BufferSize)] LPBYTE *Buffer,
   [out, ref] LPDWORD BufferSize
   );

error_status_t
FAX SetJob(
   [in] handle t hBinding,
   [in] DWORD JobId,
   [in] DWORD Command
   );

error status t
FAX GetPageData(
   [in] handle t hBinding,
   [in] DWORD JobId,
   [out, size_is(,*BufferSize)] LPBYTE *Buffer,
   [out, ref] LPDWORD BufferSize,
   [in, out] LPDWORD ImageWidth,
   [in, out] LPDWORD ImageHeight
   );

error status t
FAX_GetDeviceStatus(
   [in] RPC_FAX_PORT_HANDLE FaxPortHandle,
   [out, size is(,*BufferSize)] LPBYTE*StatusBuffer,
   [out, ref] LPDWORD BufferSize
```

```
   );

error_status_t
FAX Abort(
   [in] handle t hBinding,
   [in] DWORD JobId
   );

error_status_t
FAX EnumPorts(
   [in] handle t hBinding,
   [out, size is(,*BufferSize)] LPBYTE *PortBuffer,
   [out, ref] LPDWORD BufferSize,
   [out, ref] LPDWORD PortsReturned
   );

error status t
FAX GetPort(
   [in] RPC FAX PORT HANDLE FaxPortHandle,
   [out, size_is(,*BufferSize)] LPBYTE *PortBuffer,
   [out, ref] LPDWORD BufferSize
   );

error status t
FAX_SetPort(
   [in] RPC_FAX_PORT_HANDLE FaxPortHandle,
   [in] const FAX_PORT_INFO *PortInfo
   );

error status t
FAX EnumRoutingMethods(
    [in] RPC_FAX_PORT_HANDLE FaxPortHandle,
    [out, size_is(,*RoutingInfoBufferSize)] LPBYTE *RoutingInfoBuffer,
    [out, ref] LPDWORD RoutingInfoBufferSize,
    [out, ref] LPDWORD PortsReturned
    );

error_status_t
FAX_EnableRoutingMethod(
    [in] RPC FAX PORT HANDLE FaxPortHandle,
    [in, string, unique] LPCWSTR RoutingGuid,
    [in] BOOL Enabled
    );

error status t
FAX GetRoutingInfo(
    [in] RPC FAX PORT HANDLE FaxPortHandle,
    [in, string, unique] LPCWSTR RoutingGuid,
    [out, size_is(,*RoutingInfoBufferSize)] LPBYTE *RoutingInfoBuffer,
    [out, ref] LPDWORD RoutingInfoBufferSize
    );

error status t
FAX SetRoutingInfo(
   [in] RPC_FAX_PORT_HANDLE FaxPortHandle,
   [in, string, unique] LPCWSTR RoutingGuid,
   [in, unique, size is(RoutingInfoBufferSize)] const BYTE *RoutingInfoBuffer,
   [in, range(0,FAX MAX RPC BUFFER)] DWORD RoutingInfoBufferSize
   );


error_status_t
FAX_EnumGlobalRoutingInfo(
    [in] handle t hBinding,
    [out, size_is(,*RoutingInfoBufferSize)] LPBYTE *RoutingInfoBuffer,
```

```
    [out, ref] LPDWORD RoutingInfoBufferSize,
    [out, ref] LPDWORD MethodsReturned
    );

error status t
FAX SetGlobalRoutingInfo(
    [in] handle t hBinding,
    [in] const FAX_GLOBAL_ROUTING_INFOW *RoutingInfo
    );

error status t
FAX GetConfiguration(
   [in] handle t hBinding,
   [out, size_is(,*BufferSize)] LPBYTE *Buffer,
   [out, ref] LPDWORD BufferSize
   );

error status t
FAX SetConfiguration(
   [in] handle_t hBinding,
   [in] const FAX CONFIGURATIONW *FaxConfig
   );

error status t
FAX_GetLoggingCategories(
    [in] handle_t hBinding,
    [out, size_is(,*BufferSize)] LPBYTE *Buffer,
    [out, ref] LPDWORD BufferSize,
    [out, ref] LPDWORD NumberCategories
    );

error_status_t
FAX_SetLoggingCategories(
    [in] handle t hBinding,
    [in, unique, size is(BufferSize)] const LPBYTE Buffer,
    [in, range(0,FAX MAX RPC BUFFER)] DWORD BufferSize,
    [in] DWORD NumberCategories
    );

error status t
FAX GetSecurity(
    [in] handle t hBinding,
    [out, size_is(,*lpdwBufferSize)] LPBYTE * pSecurityDescriptor,
    [out, ref] LPDWORD lpdwBufferSize
    )
    ;

error status t
FAX_SetSecurity(
    [in] handle_t hBinding,
    [in] SECURITY INFORMATION SecurityInformation,
    [in, unique, size is(dwBufferSize)] const LPBYTE pSecurityDescriptor,
    [in, range(0,FAX MAX RPC BUFFER)] DWORD dwBufferSize
    );

error_status_t
FAX AccessCheck(
    [in] handle t hBinding,
    [in] DWORD AccessMask,
    [out, ref] BOOL* pfAccess,
    [in, out, unique] LPDWORD  lpdwRights
    );
```

```
error status t
FAX_CheckServerProtSeq(
    [in] handle_t hbinding,
    [in, out, unique] LPDWORD lpdwProtSeq
    );


error_status_t
FAX_SendDocumentEx
(
    [in] handle t hBinding,
    [in, string, unique] LPCWSTR lpcwstrFileName,
    [in] LPCFAX COVERPAGE INFO EXW  lpcCoverPageInfo,
    [in] LPCFAX_PERSONAL_PROFILEW   lpcSenderProfile,
    [in, range(0,FAX_MAX_RECIPIENTS) ] DWORD dwNumRecipients,
    [in, size is(dwNumRecipients)] LPCFAX PERSONAL PROFILEW     lpcRecipientList,
    [in] LPCFAX JOB PARAM EXW lpJobParams,
    [in, out, unique] LPDWORD lpdwJobId,
    [out] PDWORDLONG lpdwlMessageId,
    [out, size_is(dwNumRecipients)] PDWORDLONG lpdwlRecipientMessageIds
);

error status t
FAX EnumJobsEx(
   [in] handle_t hBinding,
   [in] DWORD dwJobTypes,
   [out, size_is(,*BufferSize)] LPBYTE *Buffer,
   [out, ref] LPDWORD BufferSize,
   [out, ref] LPDWORD lpdwJobs
   );

error_status_t
FAX_GetJobEx(
   [in] handle t hBinding,
   [in] DWORDLONG dwlMessageID,
   [out, size is(,*BufferSize)] LPBYTE *Buffer,
   [out, ref] LPDWORD BufferSize
   );


error status t
FAX GetCountryList(
    [in] handle_t    FaxHandle,
    [out, size_is(,*BufferSize)] LPBYTE *Buffer,
    [out, ref] LPDWORD BufferSize
   );


error_status_t
FAX_GetPersonalProfileInfo
(
    [in] handle t hBinding,
    [in] DWORDLONG dwlMessageId,
    [in] FAX ENUM MESSAGE FOLDER   dwFolder,
    [in] FAX_ENUM_PERSONAL_PROF_TYPES  ProfType,
    [out, size_is(,*BufferSize)] LPBYTE *Buffer,
    [out, ref] LPDWORD BufferSize
);


error status t
FAX_GetQueueStates (
    [in]  handle_t    hFaxHandle,
    [out] LPDWORD     pdwQueueStates
);
```

```
error_status_t
FAX_SetQueue (
    [in] handle t       hFaxHandle,
    [in] const DWORD    dwQueueStates
);


error_status_t
FAX GetReceiptsConfiguration (
    [in]  handle t    hFaxHandle,
    [out, size is(,*BufferSize)] LPBYTE *Buffer,
    [out, ref] LPDWORD BufferSize
);

error status t
FAX SetReceiptsConfiguration (
    [in] handle t       hFaxHandle,
    [in, ref] const PFAX RECEIPTS CONFIGW pReceipts
);

error status t
FAX GetReceiptsOptions (
    [in] handle t       hFaxHandle,
    [out, ref]  LPDWORD lpdwReceiptsOptions
);


error status t
FAX GetVersion (
    [in]  handle t      hFaxHandle,
    [in, out] PFAX_VERSION  pVersion
);


error status t
FAX GetOutboxConfiguration (
    [in]  handle_t    hFaxHandle,
    [out, size_is(,*BufferSize)] LPBYTE *Buffer,
    [out, ref] LPDWORD BufferSize
);

error_status_t
FAX_SetOutboxConfiguration (
    [in] handle t       hFaxHandle,
    [in, ref] const PFAX OUTBOX CONFIG pOutboxCfg
);

error_status_t
FAX_GetPersonalCoverPagesOption (
    [in] handle t       hFaxHandle,
    [out, ref] LPBOOL   lpbPersonalCPAllowed
);


error_status_t
FAX GetArchiveConfiguration (
    [in] handle t                                   hFaxHandle,
    [in] FAX ENUM MESSAGE FOLDER                    Folder,
    [out, size is(,*BufferSize)] LPBYTE             *Buffer,
    [out, ref] LPDWORD                              BufferSize
);

error status t
FAX_SetArchiveConfiguration (
```

```
    [in] handle_t                       hFaxHandle,
    [in] FAX_ENUM_MESSAGE_FOLDER        Folder,
    [in, ref] const PFAX_ARCHIVE_CONFIGW pArchiveCfg
);

error status_t
FAX_GetActivityLoggingConfiguration (
    [in] handle_t                                   hFaxHandle,
    [out, size_is(,*BufferSize)] LPBYTE             *Buffer,
    [out, ref] LPDWORD                              BufferSize
);

error status_t
FAX_SetActivityLoggingConfiguration (
    [in] handle_t                                   hFaxHandle,
    [in, ref] const PFAX_ACTIVITY_LOGGING_CONFIGW   pActivLogCfg
);

error status_t
FAX_EnumerateProviders (
    [in] handle_t                                   hFaxHandle,
    [out, size_is(,*BufferSize)]                    LPBYTE *Buffer,
    [out, ref] LPDWORD                              BufferSize,
    [out, ref] LPDWORD                              lpdwNumProviders
);

error_status_t
FAX_GetPortEx (
    [in] handle_t                                   hFaxHandle,
    [in] DWORD                                      dwDeviceId,
    [out, size_is(,*BufferSize)] LPBYTE             *Buffer,
    [out, ref] LPDWORD                              BufferSize
);

error status_t
FAX_SetPortEx (
    [in] handle_t                                   hFaxHandle,
    [in] DWORD                                      dwDeviceId,
    [in, ref] const PFAX_PORT_INFO_EXW              pPortInfo
);

error status_t
FAX_EnumPortsEx (
    [in] handle_t                                   hFaxHandle,
    [out, size_is(,*BufferSize)]                    LPBYTE *Buffer,
    [out, ref] LPDWORD                              BufferSize,
    [out, ref] LPDWORD                              lpdwNumPorts
);


error status_t
FAX_GetExtensionData (
    [in] handle_t                                      hFaxHandle,
    [in] DWORD                                         dwDeviceId,
    [in,string,ref] LPCWSTR                            lpctstrNameGUID,
    [out, size_is(,*lpdwDataSize)] LPBYTE              *ppData,
    [out, ref] LPDWORD                                 lpdwDataSize
);

error status_t
FAX_SetExtensionData (
    [in] handle_t                       hFaxHandle,
    [in,string] LPCWSTR                 lpcwstrComputerName,
    [in] DWORD                          dwDeviceId,
    [in,string] LPCWSTR                 lpctstrNameGUID,
```

```
    [in, ref, size is(dwDataSize)] LPBYTE   pData,
    [in,range(0,FAX_MAX_RPC_BUFFER)] DWORD    dwDataSize
);

error status t
FAX AddOutboundGroup (
    [in] handle t               hFaxHandle,
    [in, string,ref] LPCWSTR    lpwstrGroupName
    );

error status t
FAX SetOutboundGroup (
    [in] handle t                                   hFaxHandle,
    [in, ref] PRPC_FAX_OUTBOUND_ROUTING_GROUPW    pGroup
    );

error status t
FAX RemoveOutboundGroup (
    [in] handle t               hFaxHandle,
    [in, string, ref] LPCWSTR   lpwstrGroupName
    );

error status t
FAX EnumOutboundGroups (
    [in] handle_t                                           hFaxHandle,
    [out, size_is(,*lpdwDataSize)] LPBYTE                   *ppData,
    [out, ref] LPDWORD                                      lpdwDataSize,
    [out, ref] LPDWORD                                      lpdwNumGroups
    );

error status t
FAX_SetDeviceOrderInGroup (
    [in] handle_t               hFaxHandle,
    [in, string, ref] LPCWSTR   lpwstrGroupName,
    [in] DWORD                  dwDeviceId,
    [in] DWORD                  dwNewOrder
    );

error_status_t
FAX AddOutboundRule (
    [in] handle t                   hFaxHandle,
    [in] DWORD                      dwAreaCode,
    [in] DWORD                      dwCountryCode,
    [in] DWORD                      dwDeviceId,
    [in, string, unique] LPCWSTR    lpwstrGroupName,
    [in] BOOL                       bUseGroup
    );

error_status_t
FAX_RemoveOutboundRule (
    [in] handle t                   hFaxHandle,
    [in] DWORD                      dwAreaCode,
    [in] DWORD                      dwCountryCode
    );


error status t
FAX SetOutboundRule (
    [in] handle t                                   hFaxHandle,
    [in, ref] RPC FAX OUTBOUND ROUTING RULEW*       pRule
    );

error_status_t
FAX EnumOutboundRules (
    [in] handle_t                                       hFaxHandle,
```

```
    [out, size is(,*lpdwDataSize)] LPBYTE              *ppData,
    [out, ref] LPDWORD                                 lpdwDataSize,
    [out, ref] LPDWORD                                 lpdwNumRules
    );

error status t
FAX RegisterServiceProviderEx (
    [in] handle_t           hFaxHandle,
    [in,string,ref] LPCWSTR lpctstrGUID,
    [in,string,ref] LPCWSTR lpctstrFriendlyName,
    [in,string,ref] LPCWSTR lpctstrImageName,
    [in,string,ref] LPCWSTR lpctstrTspName,
    [in] DWORD              dwFSPIVersion,
    [in] DWORD              dwCapabilities
);

error status t
FAX UnregisterServiceProviderEx (
    [in] handle t             hFaxHandle,
    [in,string,ref] LPCWSTR   lpctstrGUID
);


error status t
FAX_UnregisterRoutingExtension (
    [in] handle_t           hFaxHandle,
    [in,string,ref] LPCWSTR lpctstrExtensionName
);

error status t
FAX StartMessagesEnum (
    [in] handle_t                       hFaxHandle,
    [in] FAX_ENUM_MESSAGE_FOLDER        Folder,
    [out,ref] PRPC FAX MSG ENUM HANDLE  lpHandle
);

error status t
FAX_EndMessagesEnum (
    [in,out,ref] PRPC_FAX_MSG_ENUM_HANDLE  lpHandle
);

error status t
FAX_EnumMessages(
    [in,ref] RPC_FAX_MSG_ENUM_HANDLE                   hEnum,
    [in] DWORD                                         dwNumMessages,
    [out, size is(,*lpdwBufferSize)]                   LPBYTE *lppBuffer,
    [out, ref] LPDWORD                                 lpdwBufferSize,
    [out, ref] LPDWORD                                 lpdwNumMessagesRetrieved
);

error status t
FAX GetMessage (
    [in] handle t                                      hFaxHandle,
    [in] DWORDLONG                                     dwlMessageId,
    [in] FAX_ENUM_MESSAGE_FOLDER                       Folder,
    [out, size_is(,*lpdwBufferSize)] LPBYTE            *lppBuffer,
    [out, ref] LPDWORD                                 lpdwBufferSize
);

error status t
FAX RemoveMessage (
    [in] handle_t                                      hFaxHandle,
    [in] DWORDLONG                                     dwlMessageId,
    [in] FAX ENUM MESSAGE FOLDER                       Folder
);
```

```
error_status_t
FAX_StartCopyToServer (
    [in] handle t                   hFaxHandle,
    [in,string,ref] LPCWSTR         lpcwstrFileExt,
    [in,out,string,ref] LPWSTR lpwstrServerFileName,
    [out,ref] PRPC FAX COPY HANDLE  lpHandle
);

error status t
FAX StartCopyMessageFromServer (
    [in] handle t                   hFaxHandle,
    [in] DWORDLONG                  dwlMessageId,
    [in] FAX_ENUM_MESSAGE_FOLDER    Folder,
    [out,ref] PRPC_FAX_COPY_HANDLE  lpHandle
);

error status t
FAX WriteFile (
    [in,ref] RPC_FAX_COPY_HANDLE                hCopy,
    [in,ref,size is(dwDataSize)] const LPBYTE    lpbData,
    [in, range(0,RPC COPY BUFFER SIZE)] DWORD   dwDataSize
);

error_status_t
FAX_ReadFile (
    [in,ref]    RPC_FAX_COPY_HANDLE hCopy,
    [in]  DWORD   dwMaxDataSize,
    [out,ref,size is(*lpdwDataSize)]    LPBYTE  lpbData,
    [in,out,ref]   LPRANGED DWORD lpdwDataSize
);

error_status_t
FAX EndCopy (
    [in,out,ref] PRPC FAX COPY HANDLE  lphCopy
);

error_status_t
FAX_StartServerNotification(
   [in] handle t hBinding,
   [in, string, ref] LPCWSTR        lpcwstrMachineName,
   [in, string, ref] LPCWSTR        lpcwstrEndPoint,
   [in] ULONG64                     Context,
   [in, ref, string] LPCWSTR         lpcwstrProtseqString,
   [in] BOOL                        bEventEx,
   [in] DWORD                       dwEventTypes,
   [out,ref] PRPC FAX EVENT HANDLE  lpHandle
   );

error_status_t
FAX StartServerNotificationEx(
   [in] handle t hBinding,
   [in, string, ref] LPCWSTR            lpcwstrMachineName,
   [in, string, ref] LPCWSTR            lpcwstrEndPoint,
   [in] ULONG64                         Context,
   [in, ref, string] LPCWSTR             lpcwstrProtSeq,
   [in] BOOL                            bEventEx,
   [in] DWORD                           dwEventTypes,
   [out,ref] PRPC FAX EVENT EX HANDLE   lpHandle
   );

error_status_t
FAX_EndServerNotification (
    [in,out,ref] PRPC FAX EVENT EX HANDLE  lpHandle
);
```

```
error_status_t
FAX_GetServerActivity(
    [in] handle t                   hFaxHandle,
    [in, out, ref] PFAX SERVER ACTIVITY  pServerActivity
);

error_status_t
FAX_SetConfigWizardUsed (
    [in] handle t   hFaxHandle,
    [in] BOOL       bConfigWizardUsed
);

error_status_t
FAX_EnumRoutingExtensions (
    [in] handle t                                   hFaxHandle,
    [out, size is(,*BufferSize)]                    LPBYTE *Buffer,
    [out, ref] LPDWORD                              BufferSize,
    [out, ref] LPDWORD                              lpdwNumExts
);

error status t
FAX AnswerCall(
    [in] handle t           hFaxHandle,
    [in] DWORD              dwDeviceId
    );

error status t
FAX ConnectFaxServer(
    [in] handle t                       hBinding,
    [in] DWORD                          dwClientAPIVersion,
    [out, ref] LPDWORD                  lpdwServerAPIVersion,
    [out, ref] PRPC_FAX_SVC_HANDLE      pHandle
    );

error status t
FAX GetSecurityEx(
    [in] handle_t hBinding,
    [in] SECURITY_INFORMATION SecurityInformation,
    [out, size is(,*lpdwBufferSize)] LPBYTE * pSecurityDescriptor,
    [out, ref] LPDWORD lpdwBufferSize
    );

error_status_t
FAX RefreshArchive(
    [in]  handle t                hFaxHandle,
    [in]  FAX ENUM MESSAGE FOLDER  Folder
);

error_status_t
FAX SetRecipientsLimit(
    [in] handle t hbinding,
    [in] DWORD dwRecipientsLimit
);

error_status_t
FAX GetRecipientsLimit(
    [in] handle t hbinding,
    [out, ref] LPDWORD lpdwRecipientsLimit
);

error_status_t
FAX_GetServerSKU(
    [in] handle t hbinding,
    [out, ref] PRODUCT_SKU_TYPE* pServerSKU
```

```
);

error_status_t
FAX CheckValidFaxFolder(
   [in] handle t hBinding,
   [in, string, ref] LPCWSTR        lpcwstrPath
);

error_status_t
FAX GetJobEx2(
   [in] handle t hBinding,
   [in] DWORDLONG dwlMessageID,
   [in]    DWORD      level,
   [out, size_is(,*BufferSize)] LPBYTE *Buffer,
   [out, ref] LPDWORD BufferSize
);

error status t
FAX EnumJobsEx2(
   [in] handle_t hBinding,
   [in] BOOL fAllAccounts,
   [in, string, unique] LPCWSTR lpcwstrAccountName,
   [in] DWORD dwJobTypes,
   [in]    DWORD level,
   [out, size_is(,*BufferSize)] LPBYTE *Buffer,
   [out, ref] LPDWORD BufferSize,
   [out, ref] LPDWORD lpdwJobs
);

error status t
FAX GetMessageEx (
    [in] handle_t                                    hFaxHandle,
    [in] DWORDLONG                                   dwlMessageId,
    [in] FAX ENUM MESSAGE FOLDER                     Folder,
    [in] DWORD                                       level,
    [out, size is(,*lpdwBufferSize)] LPBYTE          *lppBuffer,
    [out, ref] LPDWORD                               lpdwBufferSize
);

error status t
FAX StartMessagesEnumEx (
    [in] handle t                       hFaxHandle,
    [in] BOOL                           fAllAccounts,
    [in, string, unique] LPCWSTR        lpcwstrAccountName,
    [in] FAX ENUM MESSAGE FOLDER        Folder,
    [in] DWORD                          level,
    [out,ref] PRPC FAX MSG ENUM HANDLE  lpHandle
);

error_status_t
FAX EnumMessagesEx(
    [in,ref] RPC FAX MSG ENUM HANDLE                 hEnum,
    [in] DWORD                                       dwNumMessages,
    [out, size is(,*lpdwBufferSize)]                 LPBYTE *lppBuffer,
    [out, ref] LPDWORD                               lpdwBufferSize,
    [out, ref] LPDWORD                               lpdwNumMessagesRetrieved,
    [out, ref] LPDWORD                               lpdwLevel
);

error status t
FAX StartServerNotificationEx2(
   [in] handle_t                        hBinding,
   [in, string, unique] LPCWSTR         lpcwstrAccountName,
   [in, string, ref] LPCWSTR            lpcwstrMachineName,
   [in, string, ref] LPCWSTR            lpcwstrEndPoint,
```

```
   [in] ULONG64                          Context,
   [in, ref, string] LPCWSTR         lpcwstrProtseqString,
   [in] DWORD                            dwEventTypes,
   [in] DWORD                            level,
   [out,ref] PRPC FAX EVENT EX HANDLE    lpHandle
   );

error_status_t
FAX_CreateAccount(
    [in] handle t hBinding,
    [in] DWORD level,
    [in, ref, size is(BufferSize)] const LPBYTE   Buffer,
    [in,range(0,FAX MAX RPC BUFFER)] DWORD    BufferSize
);

error status t
FAX DeleteAccount(
    [in] handle t hBinding,
    [in, string, unique] LPCWSTR         lpcwstrAccountName
);

error status t
FAX EnumAccounts(
    [in] handle t hBinding,
    [in] DWORD level,
    [out, size_is(,*BufferSize)] LPBYTE *Buffer,
    [out, ref] LPDWORD BufferSize,
    [out, ref] LPDWORD lpdwAccounts
);

error status t
FAX_GetAccountInfo(
    [in] handle_t hBinding,
    [in, string, unique] LPCWSTR  lpcwstrAccountName,
    [in] DWORD level,
    [out, size is(,*BufferSize)] LPBYTE *Buffer,
    [out, ref] LPDWORD BufferSize
);

error status t
FAX GetGeneralConfiguration(
    [in] handle t   hBinding,
    [in] DWORD      level,
    [out, size_is(,*BufferSize)] LPBYTE *Buffer,
    [out, ref] LPDWORD BufferSize
);

error status t
FAX_SetGeneralConfiguration(
    [in] handle_t   hBinding,
    [in] DWORD      level,
    [in, ref, size is(BufferSize)] const LPBYTE Buffer,
    [in,range(0,FAX MAX RPC BUFFER)] DWORD      BufferSize
);

error_status_t
FAX GetSecurityEx2(
    [in] handle t hBinding,
    [in] SECURITY INFORMATION SecurityInformation,
    [out, size is(,*lpdwBufferSize)] LPBYTE * pSecurityDescriptor,
    [out, ref] LPDWORD lpdwBufferSize
    );

error status t
FAX_SetSecurityEx2(
```

```
    [in] handle_t hBinding,
    [in] SECURITY_INFORMATION SecurityInformation,
    [in, unique, size_is(dwBufferSize)] const LPBYTE pSecurityDescriptor,
    [in, range(0,FAX_MAX_RPC_BUFFER)] DWORD dwBufferSize
    );

error_status_t
FAX_AccessCheckEx2(
    [in] handle_t hBinding,
    [in] DWORD AccessMask,
    [out, ref] BOOL* pfAccess,
    [in, out, unique] LPDWORD  lpdwRights
    );

error_status_t
FAX_ReAssignMessage(
    [in] handle_t hBinding,
    [in] DWORDLONG dwlMessageId,
    [in, ref] PFAX_REASSIGN_INFO pReAssignInfo
    );

error_status_t
FAX_SetMessage(
    [in] handle_t                   hFaxHandle,
    [in] DWORDLONG                  dwlMessageId,
    [in] FAX_ENUM_MESSAGE_FOLDER    Folder,
    [in, ref] PFAX_MESSAGE_PROPS    lpMessageProps
);

error_status_t
FAX_GetConfigOption(
    [in]  handle_t                  hFaxHandle,
    [in]  FAX_ENUM_CONFIG_OPTION    option,
    [out] LPDWORD                   lpdwValue);

}
```

6.2 Appendix A.2: FaxClient IDL

For ease of implementation, the full IDL is provided here, where "ms-dtyp.idl" is the IDL found in [MS-DTYP] appendix A. The file "imports.idl" is the IDL defined in section 6.

```
import "ms-dtyp.idl";

interface fax_imports;
interface faxclient;


[

    uuid(6099fc12-3eff-11d0-abd0-00c04fd91a4e),

    version(3.0),

    pointer_default(unique)

]
```

```
interface faxclient
{DWORD
Dummy(
    [in]    DWORD   DummyParm);
}

[
local,
#ifdef __midl
        ms_union,
#endif // __midl
    version(1.0)
]

interface fax_imports

{
#define MIDL_PASS

DWORD
Dummy2(
    [in]    DWORD   DummyParm);

}

typedef [context_handle] void* RPC_FAX_HANDLE;
typedef [ref] RPC_FAX_HANDLE *PRPC_FAX_HANDLE;
typedef struct _FAX_EVENT {
  DWORD SizeOfStruct;
```

```
  FILETIME TimeStamp;
  DWORD  DeviceId;
  DWORD EventId;
  DWORD JobId;
} FAX_EVENT;

error_status_t
FAX_OpenConnection(
   [in] handle_t hBinding,
   [in] unsigned __int64  Context,
   [out] PRPC_FAX_HANDLE FaxHandle
   );

error_status_t
FAX_ClientEventQueue(
   [in] RPC_FAX_HANDLE  FaxPortHandle,
   [in] FAX_EVENT       FaxEvent
   );

error_status_t
FAX_CloseConnection(
   [in,out] PRPC_FAX_HANDLE FaxHandle
   );

error_status_t
FAX_ClientEventQueueEx(
   [in, ref] RPC_FAX_HANDLE  hClientContext,
   [in, ref, size_is(dwDataSize)]  const unsigned char *lpbData,
   [in] DWORD dwDataSize
```

```
);
```

7 Appendix B: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- 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 2.2.21: Structure members are aligned by the compiler at their natural boundaries. Therefore each data member is stored at an address which is a multiple of its natural size.

<2> Section 2.2.49: The FAX_EVENT_TYPE_INCOMING_CALL event is only supported in the Windows XP fax service.

<3> Section 2.2.53: The FAX_EVENT_TYPE_INCOMING_CALL event is only supported in the Windows XP fax service.

<4> Section 2.2.62: DRT_UNUSED used to be DRT_INBOX which was available in BackOffice Server 2000, Small Business Server 2000, and Windows XP.

<5> Section 2.2.62: DRT_MSGBOX is available on BackOffice Server 2000, Small Business Server 2000, Windows XP, Windows Server 2003, Small Business Server 2003.

<6> Section 3.1.4.3: Implemented in Windows Vista and Windows Server 2008.

<7> Section 3.1.4.6: Implemented in Windows Server 2003, Small Business Server 2003, Windows XP, Windows Vista, and Windows Server 2008.

<8> Section 3.1.4.7: BackOffice Server 2000, Small Business Server 2000, and Windows XP: This method supports IPX/SPX and TCP/IP as transport protocols.

Windows Server 2003: This method supports only TCP/IP as the transport protocol.

<9> Section 3.1.4.8: Implemented in Windows Server 2003, Small Business Server 2003, Windows Vista, and Windows Server 2008.

<10> Section 3.1.4.12: Implemented in Windows Vista and Windows Server 2008.

<11> Section 3.1.4.13: Implemented in Windows Vista and Windows Server 2008.

<12> Section 3.1.4.18: Implemented in Windows Vista and Windows Server 2008.

<13> Section 3.1.4.23: Implemented in Windows Vista and Windows Server 2008

<14> Section 3.1.4.25: Implemented in Windows Vista and Windows Server 2008.

<15> Section 3.1.4.31: Implemented in Windows Server 2003, Small Business Server 2003, Windows XP, Windows Vista, and Windows Server 2008.

<16> Section 3.1.4.32: Implemented in Windows Vista and Windows Server 2008.

<17> Section 3.1.4.35: Implemented in Windows Vista and Windows Server 2008

<18> Section 3.1.4.40: Implemented in Windows Vista and Windows Server 2008.

<19> Section 3.1.4.46: Implemented in Windows Vista and Windows Server 2008.

<20> Section 3.1.4.56: Implemented in Windows Vista and Windows Server 2008.

<21> Section 3.1.4.59: Implemented in Windows Server 2003, Small Business Server 2003, Windows XP, Windows Vista, and Windows Server 2008.

<22> Section 3.1.4.68: Implemented in Windows Server 2003, Small Business Server 2003, Windows XP, Windows Vista, and Windows Server 2008.

<23> Section 3.1.4.73: Fax servers provide a set of cover pages to be used by the clients. In such a case, the name of the cover page is sent on the wire during submission of faxes. Optionally, Windows-based clients can create and use their own cover pages. In this case, the format of the file is MS-EMFPLUS and is stored with a file name extension of .cov. The fax server converts the .cov file to a .TIF using standard APIs for MS-EMFPLUS.

<24> Section 3.1.4.75: Deprecated in Windows Vista and above. In this case, the server MUST return ERROR_NOT_SUPPORTED.

<25> Section 3.1.4.91: In Windows Server 2008 implementation, only e-mail receipts are valid. In previous Windows implementations, message box types of receipts were also valid.

<26> Section 3.1.4.92: Deprecated in Windows Vista and above. In this case, the server MUST return ERROR_NOT_SUPPORTED.

<27> Section 3.1.4.94: Deprecated in Windows Vista and above. In this case, the server MUST return ERROR_NOT_SUPPORTED.

<28> Section 3.1.4.95: Implemented in Windows Vista and Windows Server 2008.

<29> Section 3.1.4.95: Supported only on Windows Vista or later. For earlier versions of Windows, use FAX_SetSecurity.

<30> Section 3.1.4.99: Added in Windows Vista and above.

<31> Section 3.1.4.100: BackOffice Server 2000, Small Business Server 2000, and Windows XP

<32> Section 3.1.4.104: Implemented in Windows Server 2003, Small Business Server 2003, Windows Vista, and Windows Server 2008.

<33> Section 5.1: We try to connect three times, twice using RPC_C_AUTHN_LEVEL_PKT_PRIVACY Authentication level and if we fail we drop to RPC_C_AUTHN_LEVEL_NONE Authentication level. The **Authentication service** used is RPC_C_AUTHN_WINNT (NTLM SPP authenticator).

<34> Section 5.1: The following access control entries are used by the fax service:

1. Submit low-priority faxes: This privilege allows the user to submit low-priority fax jobs. Users can view and manage their jobs in the fax server's queue and their messages in the outgoing fax archive.

By default, this privilege is given to interactive users, everyone and administrators group.

2. Submit normal-priority faxes: This privilege allows the user to submit normal-priority fax jobs. Users can view and manage their jobs in the fax server's queue and their messages in the outgoing fax archive.

   By default, this privilege is given to interactive users, everyone and administrators group.

3. Submit high-priority faxes: This privilege allows the user to submit high-priority fax jobs. By virtue of this privilege, the user can also submit low-priority and normal-priority fax jobs. Users can view and manage their jobs in the fax server's queue and their messages in the outgoing fax archive.

   By default, this privilege is given to interactive users and administrators group.

4. View service configuration: This privilege allows the user to view and query the fax server's configuration data.

   By default, this privilege is given to interactive users and administrators group.

5. Manage service configuration: This privilege allows the user to view, and set the fax server's configuration data.

   By default, this privilege is given to interactive users and administrators group.

6. Manage server receive folder: This privilege allows the user to manage all the messages in the server's receive folder. This includes the right to reassign and delete messages.

   By default, this privilege is given to the administrator's group.

   This privilege is available only in Windows Vista, and not applicable to other versions of Windows.

7. View fax jobs: This privilege allows the user to view all outgoing jobs, including the jobs submitted by other users.

   By default, this privilege is given to the administrator's group.

8. Manage fax jobs: This privilege allows the user to manage all outgoing jobs, including the jobs submitted by other users.

   By default, this privilege is given to the administrator's group.

9. View message archives: This privilege allows the user to view all archived messages, including the archives of other users.

   By default, this privilege is given to the administrator's group.

10. Manage messages archives: This privilege allows the user to manage all archive messages, including the archives of other users.

    By default, this privilege is given to the administrator's group.

11. View outgoing messages archive: This privilege allows the user to view all outgoing archived messages, including the archives of other users.

    By default, this privilege is given to the administrator's group. This privilege is not available in Windows Vista, and is applicable to earlier versions of Windows.

12. Manage outgoing messages archive: This privilege allows the user to manage all outgoing archived messages, including the archives of other users.

    By default, this privilege is given to the administrator's group.

    This privilege is not available in Windows Vista, and is applicable to earlier versions of Windows.

13. View incoming messages archive: This privilege allows the user to view all incoming archived messages, including the archives of other users.

    By default, this privilege is given to the administrator's group. This privilege is not available in Windows Vista, and is applicable to earlier versions of Windows.

14. Manage incoming messages archive: This privilege allows the user to manage all incoming archived messages, including the archives of other users.

    By default, this privilege is given to the administrator's group.

    This privilege is not available in Windows Vista, and is applicable to earlier versions of Windows.

8 Index

G

I

L

M

N

O

P

R

S

T

U

V

W