[MS-MAIL]:
Remote Mailslot Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
12/18/2006	0.01		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	1.2.4	Editorial	Revised and edited the technical content.
09/28/2007	1.2.5	Editorial	Revised and edited the technical content.
10/23/2007	1.2.6	Editorial	Revised and edited the technical content.
11/30/2007	1.2.7	Editorial	Revised and edited the technical content.
01/25/2008	1.2.8	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction

The Remote Mailslot Protocol is a Microsoft-proprietary protocol and is a simple, unreliable, insecure, and unidirectional interprocess communications (IPC) protocol between a client and server. A mailslot server creates a mailslot, and a mailslot client writes messages to the mailslot created by the server. The server then reads these messages, thus achieving communication between the client and server. A mailslot is represented locally on the server as a file.

1.1 Glossary

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

Little-Endian
Named Pipe
NetBIOS Datagram Service

The following terms are specific to this document:

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.

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

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

[NETBEUI] IBM Corporation, "LAN Technical Reference: 802.2 and NetBIOS APIs", 1986, http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/BK8P7001/CCONTENTS

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

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

[RFC1088] McLaughlin III, L., "A Standard for the Transmission of IP Datagrams over NetBIOS Networks", RFC 1088, February 1989, http://www.ietf.org/rfc/rfc1088.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

[IPX] Microsoft Corporation, "Internetwork Packet Exchange (IPX)", http://msdn2.microsoft.com/en-us/library/ms817906.aspx

[MS-NRPC] Microsoft Corporation, "Netlogon Remote Protocol Specification", March 2007.

[NetBIOSSUFFIX] Microsoft Corporation, "NetBIOS Suffixes (16th Character of the NetBIOS Name)", http://support.microsoft.com/?id=163409

[PIPE] Microsoft Corporation, "Named Pipes", http://msdn2.microsoft.com/en-us/library/aa365590.aspx

[MSLOT] Microsoft Corporation, "Mailslots", http://msdn2.microsoft.com/en-us/library/aa365576.aspx

1.3 Protocol Overview (Synopsis)

The Remote Mailslot Protocol is a simple, unreliable, insecure, and unidirectional interprocess communications (IPC) protocol between a client and server or a group of servers using the **NetBIOS datagram service** (as specified in [RFC1001] section 17) as the transport protocol. A mailslot server creates a mailslot, and a mailslot client writes messages to the mailslot created by the server. The server then reads these messages, thus achieving communication between the client and server applications. If the server closes the mailslot, the client will no longer be able to send messages to this mailslot.

This protocol specifies a means of carrying SMB_COM_TRANSACTION messages (as specified in section 2.2.1) over a NetBIOS datagram service. The sender of the mailslot message formats the SMB_COM_TRANSACTION message and sends it as a NetBIOS datagram. This protocol is not transported over SMB.

1.4 Relationship to Other Protocols

The Remote Mailslot Protocol relies on the transport mechanisms of the NetBIOS datagram service (as specified in section 2.1).

The Remote Mailslot Protocol is used by the Netlogon Remote Protocol to locate domain controllers.

1.5 Prerequisites/Preconditions

The server MUST have a NetBIOS name registered, as described in [RFC1001] section 15. The higher-layer application that uses the Remote Mailslot Protocol MUST know the NetBIOS name of the server to which it is trying to connect. The higher-layer application MUST also know the name of the mailslot.

1.6 Applicability Statement

Remote mailslot messages are used in scenarios that require sending simple, short messages to one or more computers on the network. Neither the sender nor the receiver can expect reliable or ordered delivery of these messages.

Due to the unordered, unreliable, and unidirectional nature of the Remote Mailslot Protocol, clients that need a more robust or bidirectional communication mechanism with the server should use other, more reliable protocols such as **named pipes**, as specified in [PIPE]. Also, because the Remote Mailslot Protocol has no authentication, it is unsuitable for applications requiring a secure communication between the sender and receiver.<1>

1.7 Versioning and Capability Negotiation

The Remote Mailslot Protocol does not contain any version or capability negotiation.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

There are no standards assignments other than those implied by the use of the NetBIOS datagram service, as specified in [RFC1001] section 17.

2 Messages

The Remote Mailslot Protocol defines exactly one message: a mailslot write command.

2.1 Transport

Mailslot writes are delivered over the NetBIOS datagram service using one of the following transports:

- NetBIOS over UDP datagram service, as specified in [RFC1001] section 5.4. For this transport, the maximum allowed size for the **MailslotName** and **Databytes** fields (see section 2.2.1) of a mailslot write message is 443 bytes.
- NetBIOS over IPX, as specified in [MS-SMB] section 2.1.2.
- NetBIOS Extended User Interface, as specified in [NETBEUI].

For NetBIOS Extended User Interface and NetBIOS over IPX, the maximum size is dependent on the frame size of the underlying physical media. When using UDP as the underlying transport, the protocol implementation SHOULD restrict the maximum allowed data in the **MailslotName** and **Databytes** fields (see section 2.2.1) of a mailslot write message to no more than 443 bytes.<2>

2.2 Message Syntax

Mailslot messages MUST be encapsulated as the data portion of an SMB_COM_TRANSACTION data structure, as specified in [MS-SMB] section 2.2.12. The Transaction SMB data structure MUST be encapsulated as the payload of a datagram.

This section specifies the syntax of the Transaction SMB data structure as it applies to a mailslot write message. The byte ordering used is little-endian unless specified otherwise.

2.2.1 Mailslot Write Message

The **SMB_COM_TRANSACTION** data structure (see [MS-SMB] section 2.2.12) for a mailslot write message MUST be as follows:

Note The empty fields in the following table represent the continuation of the fields preceding it.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
SMB Header																															
...																															
...																															
...																															
...																															

...			
...			
...			
WordCount	TotalParameterCount		TotalDataCount
...	MaxParameterCount		MaxDataCount
...	MaxSetupCount	Reserved	Flags
...	Timeout		
...	Reserved2		ParameterCount
...	ParameterOffset		DataCount
...	DataOffset		SetupCount
Reserved3	Setup		
...			ByteCount
...	MailslotName (variable)		
...			
Padding (variable)			
...			
Databytes (variable)			
...			

SMB Header (32 bytes): The format of the 32 byte SMB header is specified in [MS-SMB] section 2.2.1. For a mailslot write message, the header fields MUST be set to the following values:

- Protocol: MUST be set to (0xFF, 'S', 'M', 'B'), as specified in [MS-SMB] section 2.2.1.

- Command: MUST be set to 0x25 to represent the SMB_COM_TRANSACTION, as specified in [MS-SMB] section 2.2.1.
- Flags: MUST be set to 0x18 to represent SMB_FLAGS_CASE_INSENSITIVE and SMB_FLAGS_CANONICALIZED_PATHS, as specified in [MS-SMB] section 2.2.1.

All other fields in the header MUST be formatted (as specified in [MS-SMB] section 2.2.1) and values set to zero.

WordCount (1 byte): An unsigned 8-bit integer that specifies the count of words that occur between the **WordCount** field and the **ByteCount** field. For a mailslot write request, it MUST be set to 0x11.

TotalParameterCount (2 bytes): An unsigned 16-bit integer that specifies the number of bytes in the Parameter Buffer. For a mailslot write request, it MUST be set to zero, and ignored on receipt. As this value MUST be set to zero, it indicates that no parameter array is being sent in this request.

TotalDataCount (2 bytes): An unsigned 16-bit integer that specifies the number of bytes in the **DataBytes** field. For a mailslot write request, it MUST be set to the size in bytes of the **DataBytes** field to be sent to the receiver. This value is always the same as **DataCount** for a mailslot write request.

MaxParameterCount (2 bytes): An unsigned 16-bit integer field. For a mailslot write request, it MUST be set to zero, and MUST be ignored on receipt.<3>

MaxDataCount (2 bytes): An unsigned 16-bit integer field. For a mailslot write request, it MUST be set to zero, and MUST be ignored on receipt.

MaxSetupCount (1 byte): An unsigned 8-bit integer field. For a mailslot write request, it MUST be set to zero, and MUST be ignored on receipt.

Reserved (1 byte): An unsigned 8-bit integer reserved for future use. It MUST be set to zero and ignored on receipt.

Flags (2 bytes): An unsigned 16-bit integer that MUST be set to 0x0002, and MUST be ignored on receipt.

Timeout (4 bytes): An unsigned 32-bit integer that MUST be set to zero and ignored on receipt.

Reserved2 (2 bytes): An unsigned 16-bit integer reserved for future use. It MUST be set to zero and ignored on receipt.

ParameterCount (2 bytes): An unsigned 16-bit integer that specifies the count of bytes in the Parameter Buffer of this packet. For a mailslot write request, it MUST be set to zero and ignored on receipt. Because this value MUST be set to zero, it indicates that no parameter array is being sent in this request.

ParameterOffset (2 bytes): An unsigned 16-bit integer that specifies the offset in bytes from the beginning of the SMB_COM_TRANSACTION packet to where the Parameter Buffer begins. For a mailslot write request it MUST be set to zero and ignored on receipt. As this value MUST be set to zero, it indicates that no parameter array is being sent in this request.

DataCount (2 bytes): An unsigned 16-bit integer that specifies the count of bytes in the **DataBytes** field. For a mailslot write request, it MUST be set to the size in bytes of the

DataBytes field to be sent to the receiver. This value is always the same as **TotalDataCount**.

DataOffset (2 bytes): An unsigned 16-bit integer that specifies the offset to the **DataBytes** field in this packet. For a mailslot write request, it MUST be set to the offset of the **Databytes** field from the beginning of the SMB_COM_TRANSACTION message.

SetupCount (1 byte): An unsigned 8-bit integer that MUST be set to 0x03.

Reserved3 (1 byte): An unsigned 8-bit integer reserved for future use. It MUST be set to zero and ignored on receipt.

ByteCount (2 bytes): An unsigned 16-bit integer that MUST specify the number of bytes that follow this field. For a mailslot write request, it MAY be ignored on receipt, and the number of bytes that follow this field is determined by using the values in the **DataOffset** and **DataCount** fields.<4>

MailslotName (variable): A null-terminated, case-insensitive ASCII string that denotes the name of the mailslot to which the message is being sent. It MUST be of the form "\mailslot\<name>", where <name> is the name of the mailslot. The <name> MUST be a non-empty string and names are not case sensitive. The name field MAY contain multiple directory levels, such as "\mailslot\directory\ms1", or a single level such as "\mailslot\ms1".<5>

Padding (variable): Padding data. This field MUST be large enough so that the **DataBytes** field is 32-bit aligned. To that end, this field MUST be between 0 and 3 bytes long, inclusive. The field MUST be set to zero and ignored on receipt.

Databytes (variable): Buffer containing the mailslot message to be delivered to the server. The size of the mailslot message MUST NOT exceed the maximum allowed size, as specified in section 2.1.

3 Protocol Details

The following sections specify Remote Mailslot Protocol client and server details, including abstract data models and message processing rules.

3.1 Client Details

The Remote Mailslot Protocol clients are higher-layer applications that use the mailslot protocol to send a message to the server, as described in section 3.1.4.1. Because this is an unreliable, unidirectional protocol, there is neither a connection phase nor an acknowledgment from the server for the send from the client.

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Application Writes to a Mailslot

A client application invokes the Remote Mailslot Protocol to send a mailslot message to a remote server by passing the data to be sent, the NetBIOS name of the remote server (as specified in [RFC1088]), and the mailslot name to the client protocol implementation.

On receipt of a request from a higher-layer application to send a mailslot message, the client mailslot implementation MUST package the data to be sent and the destination mailslot name in a SMB_COM_TRANSACTION data structure by filling in the various fields, as specified in section 2.2.1.

This SMB_COM_TRASACTION data structure is then sent as a NetBIOS datagram using the NetBIOS name of the remote server passed in by the client application as the destination address for the NetBIOS datagram, as specified in [RFC1001].

The Remote Mailslot Protocol does not provide for multipacket segmentation. As a result, if the invoking application specifies data that is too large to fit into a single NetBIOS datagram (this size is transport dependent; see section 2.1), the invocation MUST be failed and no data sent.<6>

3.1.5 Message Processing Events and Sequencing Rules

Multipacket segmentation is not supported by the Remote Mailslot Protocol. If the packet is too large to fit into a single NetBIOS datagram, then it must be quietly discarded.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

3.2.1 Abstract Data Model

The Remote Mailslot Protocol servers require higher-layer applications (running on the server) that use the protocol to specify individual mailslots. Each mailslot MUST be identified by an ASCII name that is unique for that server. The name MUST follow the format "\mailslot\<name>", where <name> MUST be the unique name of the mailslot for that server. Names are not case-sensitive.

The server maintains a look-up table for the list of active mailslots indexed by the mailslot name. Each active mailslot has a queue, called the message queue, associated with it. This message queue holds pending incoming mailslot messages. The length of the queue is implementation-specific.<7>

3.2.2 Timers

None.

3.2.3 Initialization

On initialization, the server MUST create an empty, active mailslot look-up table and rely on the NetBIOS datagram service to receive mailslot messages that are sent as broadcast NetBIOS datagrams. For more information on receiving broadcast NetBIOS datagrams by NetBIOS datagram service, see [RFC1001], section 17.

3.2.4 Higher-Layer Triggered Events

The Remote Mailslot Protocol server MUST expose interfaces to upper-layer applications to allow them to create, read, and close mailslots on the server, as specified in this section.

3.2.4.1 Application Creates a Mailslot

On a mailslot create request from an application running on the server, the server MUST attempt to create a mailslot by adding an entry with the specified name to its active mailslot table. If the name already exists, the server MUST fail the request.

After the mailslot is created, the server MUST create a queue and associate it with this mailslot. This queue is used to hold pending incoming mailslot messages, as specified in section 3.2.1.<8>

3.2.4.2 Application Reads from a Mailslot

When an application running on the server attempts to read messages from a specific mailslot, the server MUST first locate the specified mailslot in its active mailslot table. If the mailslot does not exist in the table, the request MUST be failed.

For read requests to an active mailslot, the server MUST read the next message from the message queue associated with the specified mailslot. If there are no messages in the queue to be read, the server MAY block the read until a message arrives on the queue.

After the message is read from the queue, the server MUST remove the message from the message queue.<9>

3.2.4.3 Application Closes a Mailslot

When an application running on the server closes an active mailslot on the server, the server MUST remove the mailslot from its active mailslot table. The server MUST remove any pending messages in the message queue and cancel any pending read operations associated with this mailslot. Once the mailslot is removed, and the pending messages are removed, the message queue and other resources associated with this mailslot are freed.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Server Receives a Mailslot Write

On receiving a mailslot write request from a client, the Mailslot server MUST validate that the received mailslot write conforms to the syntax specified in section 2.2.1. If either the format or the contents does not conform to the specified syntax, the request MUST be ignored and silently discarded by the Mailslot server.

For all valid requests, the Mailslot server MUST read the name of the mailslot from the **MailslotName** field of the request (see section 2.2.1). The Mailslot server MUST look up this name in its active mailslot table to find the associated mailslot. If an active mailslot does not exist on the Mailslot server, the request MUST be ignored and silently discarded.

For requests that have an active mailslot available on the Mailslot server, the Mailslot server MUST read the **DataOffset** field of the request (see section 2.2.1) to calculate the offset from the beginning of the SMB_COM_TRANSACTION message to the **Databytes** field (see section 2.2.1) that contains the mailslot message.

Note Only the act of adding a message to the mailslot needs to be atomic, not the entire block of operations from reading, parsing, and update.

The Mailslot server MUST read the **DataCount** field of the request (see section 2.2.1) to determine the size of the mailslot message and MUST read **DataCount** bytes from the start of the **Databytes** buffer to obtain the actual mailslot message. The Mailslot server MUST attempt to atomically add the actual mailslot to the message queue of the associated mailslot. If the message cannot be added to the message queue for any reason, the Mailslot server MUST silently discard the request.

Any messages arriving after the mailslot has been closed MUST be ignored and silently discarded by the Mailslot server.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

Mailslots are supported by three higher-level specialized functions, CreateMailslot, GetMailslotInfo, and SetMailslotInfo. These functions are used by the mailslot server. Note that none of these translate into SMB commands, as specified in [MS-SMB] section 2.2.1.

CreateMailslot should create a local mailslot and return the server-side handle to this mailslot, and map to the higher-level server-side event (see section 3.2.4.1). GetMailslotInfo and SetMailslotInfo are server-specific configurations about how long the server will block a read request waiting for the message to arrive before failing back to the application, and it is implementation-specific API for local operations and not related to how the protocol functions.

The client would call CreateFile to open a mailslot, and then call WriteFile (for more information, see [MSLOT]) to write to it. It is this write file call that generates traffic, as specified in section 3.1.4.1.

The following network traffic capture depicts the protocol message sequence for a mailslot write from a client to a mailslot server with the mailslot name \MAILSLOT\test1\sample_mailslot. It is generated when a client application invokes the Remote Mailslot Protocol to send a mailslot message to a remote server, as specified in section 3.1.4.1. For a detailed description of the fields in this example, see section 2.2.1.

FRAME 1 – SMB_COM_TRANSACTION MailSlot Write

```
Unparsed:
FF 53 4D 42 25 00 00 00 00 18 04 00 00 00 00 00
     SMB%............
00 00 00 00 00 00 00 00 00 00 FF FE 00 00 00 00
     ..........ÿþ.....
11 00 00 24 00 02 00 00 00 00 00 02 00 00 00 00
     ..$.............
00 00 00 00 00 68 00 24 00 68 00 03 00 01 00 00
     .....h.$.h.......
00 02 00 47 00 5C 4D 41 49 4C 53 4C 4F 54 5C 74
     ..G.\MAILSLOT\te
65 73 74 31 5C 73 61 6D 70 6C 65 5F 6D 61 69 6C
     st1\sample_mails
73 6C 6F 74 00 00 00 00 CA CA CA CA CA CA CA CA
     lot....ÊÊÊÊÊÊÊÊ
CA CA CA CA CA CA CA CA CA CA CA CA CA CA CA CA
     ÊÊÊÊÊÊÊÊÊÊÊÊÊÊÊÊ
CA CA CA CA CA CA CA CA CA CA CA CA
     ÊÊÊÊÊÊÊÊÊÊÊÊ

Parsed:
Smb: C; Transact, Mail Slots, Write Mail Slot,
 FileName = \MAILSLOT\test1\sample_mailslot
Protocol: SMB
Command: Transact 37(0x25)
DOSError: No Error
ErrorClass: No Error
Reserved: 0 (0x0)
Error: No Error
SMBHeader: Command, TID: 0x0000, PID: 0xFEFF,
UID: 0x0000, MID: 0x0000
Flags: 24 (0x18)
Flags2: 4 (0x4)
```

```
PIDHigh: 0 (0x0)
SecuritySignature: 0x0
Reserved: 0 (0x0)
TreeID: 0 (0x0)
ProcessID: 65279 (0xFEFF)
UserID: 0 (0x0)
MultiplexID: 0 (0x0)
CTransaction:
WordCount: 17 (0x11)
TotalParameterCount: 0 (0x0)
TotalDataCount: 36 (0x24)
MaxParameterCount: 2 (0x2)
MaxDataCount: 0 (0x0)
MaxSetupCount: 0 (0x0)
Reserved1: 0 (0x0)
Flags: Do not disconnect TID
Timeout: 0 sec(s)
Reserved2: 0 (0x0)
ParameterCount: 0 (0x0)
ParameterOffset: 104 (0x68)
DataCount: 36 (0x24)
DataOffset: 104 (0x68)
SetupCount: 3 (0x3)
Reserved3: 0 (0x0)
MailSlotsSetupWords:
MailSlotOpcode: Write Mail Slot
TransactionPriority: 0 (0x0)
MailSlotClass: Unreliable & Broadcast
ByteCount: 71 (0x47)
MailSlotsBuffer:
FileName: \MAILSLOT\test1\sample_mailslot
Pad2: Binary Large Object (3 Bytes)
MailSlotData: Binary Large Object (36 Bytes)
CA CA CA CA CA CA CA CA CA CA CA CA CA CA CA CA
CA CA CA CA CA CA CA CA CA CA CA CA CA CA CA CA
CA CA CA CA
```

5 Security

The following sections specify security considerations for implementers of the Remote Mailslot Protocol.

5.1 Security Considerations for Implementers

The Remote Mailslot Protocol is not a secure protocol. Do not use the Remote Mailslot Protocol if applications need secure communication between client and server.

5.2 Index of Security Parameters

None.

6 Appendix A: 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
- Windows Server 2008

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 1.6: Windows 2000 and later versions of Windows support the Remote Mailslot Protocol.

<2> Section 2.1: The use of NetBIOS over IPX or NetBEUI is deprecated and not used by default. NetBIOS over UDP is the default transport used by the Windows implementation of the Remote Mailslot Protocol. When using NetBIOS over UDP, the Windows implementation does not enforce the limit of 443 bytes for the combined **MailslotName** and **Databytes** size. As a result, the maximum allowed size is restricted only by the maximum size allowed for a UDP datagram. For Windows, this is 65,424 bytes (that is, the size of the source and destination NetBIOS names in the NetBIOS datagram, as specified in [RFC1001] section 17.1.2).

<3> Section 2.2.1: Windows implementations set this field to 0x2.

<4> Section 2.2.1: Windows implementations ignore this field.

<5> Section 2.2.1: Windows supports multiple directory levels in the mailslot name, and passes such names in the mailslot write operation if the application requests it.

<6> Section 3.1.4.1: Windows clients using the Remote Mailslot Protocol follow the NetBIOS naming conventions when specifying the NetBIOS name. For more information, see [NetBIOSSUFFIX].

<7> Section 3.2.1: There is not any enforced limit on the number of entries in the queue.

<8> Section 3.2.4.1: If a mailslot with the specified name already exists, the server fails the request to create the mailslot.

<9> Section 3.2.4.2: When there are no messages in the queue, Windows waits for a user-specified time out in milliseconds. If the specified time out is zero, the read returns immediately with a specific failure error code that would indicate an empty message queue.

7 Index

V

W