

[MC-IISIAQ]: Internet Information Services (IIS) IAQ AdminRPC Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: <http://www.microsoft.com/interop/osp>) or the Community Promise (available here: <http://www.microsoft.com/interop/cp/default.mspx>). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
08/10/2007	0.1	Major	Initial Availability
09/28/2007	0.2	Minor	Updated the technical content.
10/23/2007	0.3	Minor	Updated the technical content.
11/30/2007	0.3.1	Editorial	Revised and edited the technical content.
01/25/2008	0.3.2	Editorial	Revised and edited the technical content.
03/14/2008	0.3.3	Editorial	Revised and edited the technical content.
05/16/2008	0.3.4	Editorial	Revised and edited the technical content.
06/20/2008	0.4	Minor	Updated the technical content.
07/25/2008	0.4.1	Editorial	Revised and edited the technical content.
08/29/2008	0.4.2	Editorial	Revised and edited the technical content.
10/24/2008	0.4.3	Editorial	Revised and edited the technical content.
12/05/2008	0.4.3	Editorial	Revised and edited the technical content.
01/16/2009	0.4.4	Editorial	Revised and edited the technical content.
02/27/2009	0.4.5	Editorial	Revised and edited the technical content.
04/10/2009	0.4.6	Editorial	Revised and edited the technical content.
05/22/2009	0.4.7	Editorial	Revised and edited the technical content.
07/02/2009	0.4.8	Editorial	Revised and edited the technical content.
08/14/2009	0.5	Minor	Updated the technical content.
09/25/2009	0.6	Minor	Updated the technical content.
11/06/2009	0.6.1	Editorial	Revised and edited the technical content.
12/18/2009	0.6.2	Editorial	Revised and edited the technical content.
01/29/2010	0.6.3	Editorial	Revised and edited the technical content.
03/12/2010	0.6.4	Editorial	Revised and edited the technical content.
04/23/2010	0.6.5	Editorial	Revised and edited the technical content.
06/04/2010	1.0	Major	Updated and revised the technical content.
07/16/2010	2.0	Major	Significantly changed the technical content.

Date	Revision History	Revision Class	Comments
08/27/2010	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	2.0	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction	7
1.1 Glossary	7
1.2 References	8
1.2.1 Normative References	8
1.2.2 Informative References	8
1.3 Overview	8
1.4 Relationship to Other Protocols	9
1.5 Prerequisites/Preconditions	9
1.6 Applicability Statement	9
1.7 Versioning and Capability Negotiation	9
1.8 Vendor-Extensible Fields	9
1.9 Standards Assignments	9
2 Messages	10
2.1 Transport	10
2.2 Common Data Types	10
2.2.1 QUEUE_ADMIN_VERSIONS	10
2.2.2 MESSAGE_FILTER	10
2.2.3 MESSAGE_FILTER_FLAGS	11
2.2.4 MESSAGE_ACTION	12
2.2.5 MESSAGE_ENUM_FILTER_TYPE	12
2.2.6 MESSAGE_ENUM_FILTER	13
2.2.7 LINK_INFO_FLAGS	14
2.2.8 LINK_ACTION	15
2.2.9 LINK_INFO	15
2.2.10 AQ_MESSAGE_FLAGS	16
2.2.11 QUEUE_INFO	16
2.2.12 MESSAGE_INFO	17
2.2.13 QUEUELINK_TYPE	18
2.2.14 QUEUELINK_ID	18
3 Protocol Details	20
3.1 Common Details	20
3.1.1 Abstract Data Model	20
3.1.2 Timers	20
3.1.3 Initialization	20
3.1.4 Message Processing Events and Sequencing Rules	20
3.1.5 Timer Events	21
3.1.6 Other Local Events	21
3.2 IAQAdmin	21
3.2.1 Abstract Data Model	21
3.2.2 Timers	21
3.2.3 Initialization	21
3.2.4 Message Processing Events and Sequence Rules	21
3.2.4.1 GetVirtualServerAdminITF (Opnum 3)	21
3.2.5 Timer Events	22
3.2.6 Other Local Events	22
3.3 IVSAQAdmin	22
3.3.1 Abstract Data Model	23
3.3.2 Timers	23

3.3.3	Initialization	23
3.3.4	Message Processing Events and Sequencing Rules.....	23
3.3.4.1	GetLinkEnum (Opnum 3)	23
3.3.4.2	StopAllLinks (Opnum 4).....	24
3.3.4.3	StartAllLinks (Opnum 5)	24
3.3.4.4	GetGlobalLinkState (Opnum 6).....	25
3.3.5	Timer Events	25
3.3.6	Other Local Events	25
3.4	IEnumVSAQLinks.....	26
3.4.1	Abstract Data Model	26
3.4.2	Timers	26
3.4.3	Initialization	26
3.4.4	Message Processing Events and Sequencing Rules.....	26
3.4.4.1	Next (Opnum 3)	26
3.4.4.2	Skip (Opnum 4).....	27
3.4.4.3	Reset (Opnum 5).....	27
3.4.4.4	Clone (Opnum 6).....	28
3.4.5	Timer Events	28
3.4.6	Other Local Events	28
3.5	IVSAQLink	29
3.5.1	Abstract Data Model	29
3.5.2	Timers	29
3.5.3	Initialization	29
3.5.4	Message Processing Events and Sequencing Rules.....	29
3.5.4.1	GetInfo (Opnum 3)	29
3.5.4.2	SetLinkState (Opnum 4).....	30
3.5.4.3	GetQueueEnum (Opnum 5).....	31
3.5.5	Timer Events	31
3.5.6	Other Local Events	31
3.6	IEnumLinkQueues.....	31
3.6.1	Abstract Data Model	32
3.6.2	Timers	32
3.6.3	Initialization	32
3.6.4	Message Processing Events and Sequencing Rules.....	32
3.6.4.1	Next (Opnum 3)	32
3.6.4.2	Skip (Opnum 4).....	33
3.6.4.3	Reset (Opnum 5).....	33
3.6.4.4	Clone (Opnum 6).....	34
3.6.5	Timer Events	34
3.6.6	Other Local Events	34
3.7	ILinkQueue	34
3.7.1	Abstract Data Model	35
3.7.2	Timers	35
3.7.3	Initialization	35
3.7.4	Message Processing Events and Sequencing Rules.....	35
3.7.4.1	GetInfo (Opnum 3)	35
3.7.4.2	GetMessageEnum (Opnum 4).....	36
3.7.5	Timer Events	36
3.7.6	Other Local Events	37
3.8	IAQEnumMessages	37
3.8.1	Abstract Data Model	37
3.8.2	Timers	37
3.8.3	Initialization	37

3.8.4	Message Processing Events and Sequencing Rules	37
3.8.4.1	Next (Opnum 3)	37
3.8.4.2	Skip (Opnum 4)	38
3.8.4.3	Reset (Opnum 5)	39
3.8.4.4	Clone (Opnum 6)	39
3.8.5	Timer Events	40
3.8.6	Other Local Events	40
3.9	IAQMessage	40
3.9.1	Abstract Data Model	40
3.9.2	Timers	40
3.9.3	Initialization	40
3.9.4	Message Processing Events and Sequencing Rules	40
3.9.4.1	GetInfo (Opnum 3)	40
3.9.4.2	GetContentStream (Opnum 4)	41
3.9.5	Timer Events	42
3.9.6	Other Local Events	42
3.10	IAQMessageAction	42
3.10.1	Abstract Data Model	43
3.10.2	Timers	43
3.10.3	Initialization	43
3.10.4	Message Processing Events and Sequencing Rules	43
3.10.4.1	ApplyActionToMessages (Opnum 3)	43
3.10.4.2	QuerySupportedActions (Opnum 4)	44
3.10.5	Timer Events	45
3.10.6	Other Local Events	45
3.11	IUniqueId	45
3.11.1	Abstract Data Model	45
3.11.2	Timers	45
3.11.3	Initialization	45
3.11.4	Message Processing Events and Sequencing Rules	45
3.11.4.1	GetUniqueId (Opnum 3)	45
3.11.5	Timer Events	46
3.11.6	Other Local Events	46
4	Protocol Examples	47
5	Security	51
5.1	Security Considerations for Implementers	51
5.2	Index of Security Parameters	51
6	Appendix A: Full IDL	52
7	Appendix B: Product Behavior	62
8	Change Tracking	63
9	Index	64

1 Introduction

This document specifies the Internet Information Services (IIS) IAQ AdminRPC Protocol, for querying and managing a **Simple Mail Transfer Protocol (SMTP)** virtual **server** with advanced queuing.

1.1 Glossary

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

client
class identifier (CLSID)
domain name (2)
globally unique identifier (GUID)
Interface Definition Language (IDL)
Microsoft Interface Definition Language (MIDL)
Network Data Representation (NDR)
opnum
remote procedure call (RPC)
server
Simple Mail Transfer Protocol (SMTP)
Unicode (UCS-2)
universally unique identifier (UUID)

The following terms are specific to this document:

cursor: A remembered position in a sequence or list, at which activity takes place.

DS: An Active Directory service or **server**.

freeze: An administrative action to halt the operation of a **link**, **queue**, or message.

link: A collection of **queues**, the messages within which have all been assigned to the same **next hop**.

next hop: A determination of the next relay action to be taken in routing the message.

next hop server: The **server** targeted by the **next hop**.

queue: A collection that holds all messages bound for the same delivery point.

proxy address: The unique designation of a mail recipient in the format required by the messaging protocol. Thus, a user may have one or more X.400 **proxy addresses**, one or more XMTP **proxy addresses**, and one or more Lotus **proxy addresses**.

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

1.2 References

1.2.1 Normative References

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

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

[MS-DCOM] Microsoft Corporation, "[Distributed Component Object Model \(DCOM\) Remote Protocol Specification](#)", March 2007.

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

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

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

[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-COM] Microsoft Corporation, "Component Object Model", <http://msdn.microsoft.com/en-us/library/aa286559.aspx>

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

1.3 Overview

This protocol defines an interface for modeling and managing Simple Mail Transfer Protocol (SMTP) servers that implement advanced queuing.

The Internet Information Services (IIS) IAQ AdminRPC Protocol is designed to perform **queue** management operations. It is also used to administer messages in the queues and **links**. (A link is a group of queues that are destined for the same **next hop server**.)

Through the IIS IAQ AdminRPC Protocol, the user can select the virtual server, enumerate the queues, enumerate all messages in the queues, and perform such actions as **freeze**, unfreeze, delete, and retrieve the content stream of the messages.

The IIS IAQ AdminRPC Protocol can also be used to enumerate all the links in the virtual server, and to perform start and stop actions on all active outbound links. It is also used to get the current state of the links.

The **client** side of this protocol is not concerned with authentication, states, or events. All of these are handled by lower-level providers.

1.4 Relationship to Other Protocols

The IIS IAQ AdminRPC Protocol uses **RPC**, as specified in [\[MS-RPCE\]](#), for underlying network authentication, event notifications, data transfer, and state management. The IIS IAQ AdminRPC Protocol is built on top of the DCOM Remote Protocol [\[MS-DCOM\]](#).

1.5 Prerequisites/Preconditions

The IIS IAQ AdminRPC Protocol depends on the RPC protocol layer for transport. It assumes the presence of an API with underlying RPC functionality.

1.6 Applicability Statement

This protocol is most appropriate for managing an advanced SMTP server implemented according to the Microsoft Exchange model with virtual servers, queues, links, and messages.

1.7 Versioning and Capability Negotiation

This protocol contains versioning of some of the structures to enable independent evolution of client and server protocol engines. The underlying RPC may have versioning/negotiation, but that aspect must be transparent to the users of this interface.

1.8 Vendor-Extensible Fields

This protocol uses HRESULT values as defined in [\[MS-ERREF\]](#) section 2.1. Vendors can define their own HRESULT values, provided that they set the C bit (0x20000000) for each vendor-defined value, indicating that the value is a customer code.

This protocol uses Win32 error codes. These values are taken from the Windows error number space defined in [\[MS-ERREF\]](#) section 2.2. Vendors SHOULD reuse those values with their indicated meanings. Choosing any other value runs the risk of a collision in the future.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

All IIS IAQ AdminRPC Protocol messages are transported via the DCOM Remote Protocol, as specified in [\[MS-DCOM\]](#).

2.2 Common Data Types

In addition to RPC base types and definitions as specified in [\[C706\]](#) and [\[MS-RPCE\]](#), additional data types are defined in this section.

This protocol also uses the following types, as specified in [\[MS-DTYP\]](#).

Type	Reference
DWORD	As specified in [MS-DTYP] section 2.2.9
GUID	As specified in [MS-DTYP] section 2.3.2
HRESULT	As specified in [MS-DTYP] section 2.2.18
LPCWSTR	As specified in [MS-DTYP] section 2.2.33
LPWSTR	As specified in [MS-DTYP] section 2.2.35
SYSTEMTIME	As specified in [MS-DTYP] section 2.3.11
ULARGE_INTEGER	As specified in [MS-DTYP] section 2.3.13
ULONG	As specified in [MS-DTYP] section 2.2.50
WCHAR	As specified in [MS-DTYP] section 2.2.59

2.2.1 QUEUE_ADMIN_VERSIONS

The **QUEUE_ADMIN_VERSIONS** enumeration specifies current and supported queue administrative versions.

```
typedef enum tagQUEUE_ADMIN_VERSIONS
{
    CURRENT_QUEUE_ADMIN_VERSION = 4
} QUEUE_ADMIN_VERSIONS;
```

CURRENT_QUEUE_ADMIN_VERSION: The current queue administrative version. All structures should have this value in their **dwVersion** fields.

2.2.2 MESSAGE_FILTER

The **MESSAGE_FILTER** structure specifies criteria for the selection of messages based on the values in the filter.

```
typedef struct tagMESSAGE_FILTER {
    DWORD dwVersion;
    DWORD fFlags;
```

```

    [string] LPCWSTR szMessageId;
    [string] LPCWSTR szMessageSender;
    [string] LPCWSTR szMessageRecipient;
    DWORD dwLargerThanSize;
    SYSTEMTIME stOlderThan;
} MESSAGE_FILTER,
*PMESSAGE_FILTER;

```

dwVersion: Version of the **MESSAGE_FILTER** structure. It MUST be **CURRENT_QUEUE_ADMIN_VERSION**.

fFlags: MUST be a valid selection of flags defined by the [MESSAGE_FILTER_FLAGS](#) enumeration, indicating which fields of the filter are specified.

szMessageId: MUST be a null-terminated **Unicode (UCS-2)** string specifying a message ID.

szMessageSender: MUST be a null-terminated ASCII string specifying the message sender to be matched.

szMessageRecipient: MUST be a null-terminated ASCII string specifying the message recipient to be matched.

dwLargerThanSize: Integer specifying the minimum size for messages selected.

stOlderThan: [SYSTEMTIME](#) structure indicating the minimum age for messages selected.

2.2.3 MESSAGE_FILTER_FLAGS

The **MESSAGE_FILTER_FLAGS** enumeration specifies bitflags that indicate which fields are in use in the [MESSAGE_FILTER](#) structure. More than one flag may be used in a value.

```

typedef enum tagMESSAGE_FILTER_FLAGS
{
    MF_MESSAGEID = 0x00000001,
    MF_SENDER = 0x00000002,
    MF_RECIPIENT = 0x00000004,
    MF_SIZE = 0x00000008,
    MF_TIME = 0x00000010,
    MF_FROZEN = 0x00000020,
    MF_FAILED = 0x00000100,
    MF_ALL = 0x40000000,
    MF_INVERTSENSE = 0x80000000
} MESSAGE_FILTER_FLAGS;

```

MF_MESSAGEID: The **MESSAGE_FILTER.szMessageId** field is specified.

MF_SENDER: The **MESSAGE_FILTER.szMessageSender** field is specified.

MF_RECIPIENT: The **MESSAGE_FILTER.szMessageRecipient** field is specified.

MF_SIZE: The **MESSAGE_FILTER.dwLargerThanSize** field is specified.

MF_TIME: The **MESSAGE_FILTER.stOlderThan** field is specified.

MF_FROZEN: The **MESSAGE_FILTER.fFrozen** field is specified.

MF_FAILED: Selects messages that have had at least one failed delivery attempt.

MF_ALL: Selects all messages.

MF_INVERTSENSE: If set, indicates the complement of the filter definition.

2.2.4 MESSAGE_ACTION

The **MESSAGE_ACTION** enumeration specifies possible administrative actions that may be applied to messages in a virtual server, link, or queue.

```
typedef enum tagMESSAGE_ACTION
{
    MA_THAW_GLOBAL = 0x00000001,
    MA_COUNT = 0x00000002,
    MA_FREEZE_GLOBAL = 0x00000004,
    MA_DELETE = 0x00000008,
    MA_DELETE_SILENT = 0x00000010
} MESSAGE_ACTION;
```

MA_THAW_GLOBAL: Unfreezes the message in the virtual server, link, or queue.

MA_COUNT: Null operation; does not affect messages but does return count.

MA_FREEZE_GLOBAL: Freezes the message in the virtual server, link, or queue.

MA_DELETE: Removes the message from the virtual server, link, or queue.

MA_DELETE_SILENT: Removes the message without generating a **Network Data Representation (NDR)**.

In certain operations, these flags may be combined.

2.2.5 MESSAGE_ENUM_FILTER_TYPE

The **MESSAGE_ENUM_FILTER_TYPE** enumeration specifies the type of filter requested. These are bitflags and can be OR'd together.

```
typedef enum tagMESSAGE_ENUM_FILTER_TYPE
{
    MEF_FIRST_N_MESSAGES = 0x00000001,
    MEF_SENDER = 0x00000002,
    MEF_RECIPIENT = 0x00000004,
    MEF_LARGER_THAN = 0x00000008,
    MEF_OLDER_THAN = 0x00000010,
    MEF_FROZEN = 0x00000020,
    MEF_N_LARGEST_MESSAGES = 0x00000040,
    MEF_N_OLDEST_MESSAGES = 0x00000080,
    MEF_FAILED = 0x00000100,
    MEF_ALL = 0x40000000,
    MEF_INVERTSENSE = 0x80000000
} MESSAGE_ENUM_FILTER_TYPE;
```

MEF_FIRST_N_MESSAGES: Returns the first MESSAGE_ENUM_FILTER.cMessages messages.

MEF_SENDER: The MESSAGE_ENUM_FILTER.szMessageSender field is specified.

MEF_RECIPIENT: The MESSAGE_ENUM_FILTER.szMessageRecipient field is specified.

MEF_LARGER_THAN: Returns messages larger than the number of bytes given in the MESSAGE_ENUM_FILTER.cbSize field.

MEF_OLDER_THAN: Returns messages older than the value given in the MESSAGE_ENUM_FILTER.stDate field.

MEF_FROZEN: Returns messages that are frozen.

MEF_N_LARGEST_MESSAGES: Returns the largest messages in the MESSAGE_ENUM_FILTER.cMessages field.

MEF_N_OLDEST_MESSAGES: Returns the oldest messages in the MESSAGE_ENUM_FILTER.cMessages field.

MEF_FAILED: Returns only messages that have had failed delivery attempts.

MEF_ALL: Selects all messages.

MEF_INVERTSENSE: Inverts the meaning of the filter.

2.2.6 MESSAGE_ENUM_FILTER

The **MESSAGE_ENUM_FILTER** structure defines criteria for enumerating messages.

```
typedef struct tagMESSAGE_ENUM_FILTER {
    DWORD dwVersion;
    DWORD mefType;
    DWORD cMessages;
    DWORD cbSize;
    DWORD cSkipMessages;
    SYSTEMTIME stDate;
    [string] LPCWSTR szMessageSender;
    [string] LPCWSTR szMessageRecipient;
} MESSAGE_ENUM_FILTER,
*PMESSAGE_ENUM_FILTER;
```

dwVersion: The version of the filter. It MUST be CURRENT_QUEUE_ADMIN_VERSION.

mefType: The [MESSAGE_ENUM_FILTER_TYPE](#) flags for the filter.

cMessages: The number of messages to return.

cbSize: The size parameter of messages.

cSkipMessages: The number of messages at the front of the queue to skip. This is provided to allow "paged" queries to the server.

stDate: The date/time parameter of messages.

szMessageSender: Messages sent by this sender match.

szMessageRecipient: Messages sent to this recipient match.

2.2.7 LINK_INFO_FLAGS

The **LINK_INFO_FLAGS** enumeration specifies the state of the link.

```
typedef enum tagLINK_INFO_FLAGS
{
    LI_ACTIVE = 0x00000001,
    LI_READY = 0x00000002,
    LI_RETRY = 0x00000004,
    LI_SCHEDULED = 0x00000008,
    LI_REMOTE = 0x00000010,
    LI_FROZEN = 0x00000020,
    LI_TYPE_REMOTE_DELIVERY = 0x00000100,
    LI_TYPE_LOCAL_DELIVERY = 0x00000200,
    LI_TYPE_PENDING_ROUTING = 0x00000400,
    LI_TYPE_PENDING_CAT = 0x00000800,
    LI_TYPE_CURRENTLY_UNREACHABLE = 0x00001000,
    LI_TYPE_DEFERRED_DELIVERY = 0x00002000,
    LI_TYPE_INTERNAL = 0x00004000,
    LI_TYPE_PENDING_SUBMIT = 0x00008000
} LINK_INFO_FLAGS;
```

LI_ACTIVE: Link has an active connection that is transferring mail.

LI_READY: Link is ready for a connection, but there are no connections.

LI_RETRY: Link is waiting for the retry interval to elapse.

LI_SCHEDULED: Link is waiting for the next scheduled time.

LI_REMOTE: Link is to be activated by remote server. A connection will not be made unless requested by a remote server.

LI_FROZEN: Link was frozen by administrative action.

LI_TYPE_REMOTE_DELIVERY: Messages on this link are being delivered remotely. This is the default type of link.

LI_TYPE_LOCAL_DELIVERY: Messages on this link are being delivered locally.

LI_TYPE_PENDING_ROUTING: Messages on this link have not been routed to their **next hop**.

LI_TYPE_PENDING_CAT: Messages on this link are pending message categorization.

LI_TYPE_CURRENTLY_UNREACHABLE: Messages on this link do not have an available route to their final destination. This is due to transient network or server errors. These messages will be retried when a route becomes available.

LI_TYPE_DEFERRED_DELIVERY: All messages in this link are delayed by originator request until a specified time.

LI_TYPE_INTERNAL: This link is an internal link not described by the previous flags.

LI_TYPE_PENDING_SUBMIT: Messages in this link have not yet been through categorization. This is the default queue that all messages pass through prior to sorting.

2.2.8 LINK_ACTION

The **LINK_ACTION** enumeration specifies possible administrative action that may be applied to links in a virtual server.

```
typedef enum tagLINK_ACTION
{
    LA_INTERNAL = 0x00000000,
    LA_KICK = 0x00000001,
    LA_FREEZE = 0x00000020,
    LA_THAW = 0x00000040
} LINK_ACTION;
```

LA_INTERNAL: For internal use only.

LA_KICK: Forces a connection to be made for this link. This will even work for connections pending retry or a scheduled connection.

LA_FREEZE: Prohibits the link from creating outbound connections.

LA_THAW: Removes the prohibition from creating outbound connections.

2.2.9 LINK_INFO

The **LINK_INFO** structure describes the state of the link in a virtual server.

```
typedef struct tagLINK_INFO {
    DWORD dwVersion;
    [string] LPWSTR szLinkName;
    DWORD cMessages;
    DWORD fStateFlags;
    SYSTEMTIME stNextScheduledConnection;
    SYSTEMTIME stOldestMessage;
    ULARGE_INTEGER cbLinkVolume;
    [string] LPWSTR szLinkDN;
    [string] LPWSTR szExtendedStateInfo;
    DWORD dwSupportedLinkActions;
} LINK_INFO,
*PLINK_INFO;
```

dwVersion: Version of the **LINK_INFO** structure. It MUST be **CURRENT_QUEUE_ADMIN_VERSION**.

szLinkName: A unique null-terminated Unicode (UCS-2) string that is the name of the link, which might be the name of the next hop.

cMessages: The number of messages queued in this link.

fStateFlags: MUST be values from the [LINK_INFO_FLAGS](#) enumeration indicating the link state.

stNextScheduledConnection: The time at which the next connection to the next hop will be attempted.

stOldestMessage: The time of the oldest message on this link.

cbLinkVolume: The total number of bytes in all messages in this link.

szLinkDN: A null-terminated Unicode (UCS-2) string containing the domain name associated with this link. It MAY be NULL, indicating the local domain.

szExtendedStateInfo: A null-terminated Unicode (UCS-2) string that provides additional information about the link state. For example, this member may contain an explanation when the link is in a retry state. It MAY be NULL.

dwSupportedLinkActions: An unsigned integer that gives a list of actions supported on this link. Possible values are defined in [LINK_ACTION](#).

2.2.10 AQ_MESSAGE_FLAGS

The **AQ_MESSAGE_FLAGS** enumeration describes message properties.

```
typedef enum tagAQ_MESSAGE_FLAGS
{
    MP_HIGH = 0x00000001,
    MP_NORMAL = 0x00000002,
    MP_LOW = 0x00000004,
    MP_MSG_FROZEN = 0x00000008,
    MP_MSG_RETRY = 0x00000010,
    MP_MSG_CONTENT_AVAILABLE = 0x00000020
} AQ_MESSAGE_FLAGS;
```

MP_HIGH: The message property High Priority has been requested by the originator.

MP_NORMAL: The message property Normal Priority has been requested by the originator.

MP_LOW: The message property Low Priority has been requested by the originator.

MP_MSG_FROZEN: The message has been frozen by the administrator.

MP_MSG_RETRY: Delivery of this message has been attempted and failed at least once.

MP_MSG_CONTENT_AVAILABLE: The content for this message can be accessed through the queue API.

2.2.11 QUEUE_INFO

The **QUEUE_INFO** structure describes the state of the queue in a link for a virtual server.

```
typedef struct tagQUEUE_INFO {
    DWORD dwVersion;
    [string] LPWSTR szQueueName;
    [string] LPWSTR szLinkName;
    DWORD cMessages;
    ULARGE_INTEGER cbQueueVolume;
    DWORD dwMsgEnumFlagsSupported;
} QUEUE_INFO,
*PQUEUE_INFO;
```

dwVersion: Version of the **QUEUE_INFO** structure. It MUST be **CURRENT_QUEUE_ADMIN_VERSION**.

szQueueName: A null-terminated Unicode (UCS-2) string that is the name of a queue.

szLinkName: A null-terminated Unicode (UCS-2) string that is the name of a link that is servicing this queue.

cMessages: Number of messages in this queue.

cbQueueVolume: Total number of bytes for all messages in the queue.

dwMsgEnumFlagsSupported: The types of message enumeration supported.

2.2.12 MESSAGE_INFO

The **MESSAGE_INFO** structure defines the details of a single mail message.

```
typedef struct tagMESSAGE_INFO {
    DWORD dwVersion;
    [string] LPWSTR szMessageId;
    [string] LPWSTR szSender;
    [string] LPWSTR szSubject;
    DWORD cRecipients;
    [string] LPWSTR szRecipients;
    DWORD cCCRecipients;
    [string] LPWSTR szCCRecipients;
    DWORD cBCCRecipients;
    [string] LPWSTR szBCCRecipients;
    DWORD fMsgFlags;
    DWORD cbMessageSize;
    SYSTEMTIME stSubmission;
    SYSTEMTIME stReceived;
    SYSTEMTIME stExpiry;
    DWORD cFailures;
    DWORD cEnvRecipients;
    DWORD cbEnvRecipients;
    [size_is(cbEnvRecipients/sizeof(WCHAR))]
    WCHAR* mszEnvRecipients;
} MESSAGE_INFO,
*PMESSAGE_INFO;
```

dwVersion: Version of the **MESSAGE_INFO** structure. It MUST be **CURRENT_QUEUE_ADMIN_VERSION**.

szMessageId: A null-terminated Unicode (UCS-2) string that is the message ID.

szSender: A null-terminated Unicode (UCS-2) string that is the sender address, from the "From:" header.

szSubject: A null-terminated Unicode (UCS-2) string that is the message subject.

cRecipients: The number of recipients named in the "To:" header.

szRecipients: A null-terminated Unicode (UCS-2) string that contains the recipient addresses from the "To:" header.

cCCRecipients: The number of recipients in the "CC:" header.

szCCRecipients: A null-terminated Unicode (UCS-2) string that contains the CC recipient addresses from the "CC:" header.

cBCCRecipients: The number of recipients in the "BCC:" header.

szBCCRecipients: A null-terminated Unicode (UCS-2) string that contains the BCC recipient addresses, from the "BCC:" header.

fMsgFlags: An [AQ_MESSAGE_FLAGS](#) enumeration describing the message properties.

cbMessageSize: The size of the message, in bytes.

stSubmission: The time the message was submitted.

stReceived: The time the message was received by this server.

stExpiry: The time the message will expire if not delivered to all recipients.

cFailures: The number of failed delivery attempts associated with the message.

cEnvRecipients: The number of envelope recipients. Envelope recipients are those to whom delivery will be attempted. Other copies of the message may reside in other queues or servers.

cbEnvRecipients: The size, in bytes, of the envelope recipients.

mszEnvRecipients: A pointer to a Unicode (UCS-2) string containing the list of envelope recipients, separated by NULL characters. The buffer itself is terminated by an additional NULL character. Each recipient string will be formatted in the **proxy address** format of 'addr-type ":" address'. The addr-type should match the address type found in the **DS** (that is, SMTP). The address should be in its native format.

2.2.13 QUEUELINK_TYPE

The **QUEUELINK_TYPE** enumeration specifies the valid types for a [QUEUELINK_ID](#) structure.

```
typedef enum tagQUEUELINK_TYPE
{
    QLT_QUEUE,
    QLT_LINK,
    QLT_NONE
} QUEUELINK_TYPE;
```

QLT_QUEUE: This value indicates that the structure is a queue.

QLT_LINK: This value indicates that the structure is a link.

QLT_NONE: This value indicates that the structure is neither a link nor a queue.

2.2.14 QUEUELINK_ID

The **QUEUELINK_ID** structure defines how a QUEUELINK structure is identified.

```
typedef struct tagQUEUELINK_ID {
    GUID uuid;
    [string] LPWSTR szName;
}
```

```
DWORD dwId;  
  QUEUELINK_TYPE qltType;  
} QUEUELINK_ID;
```

uuid: A unique [GUID](#) assigned to the structure.

szName: A null-terminated Unicode (UCS-2) string that is a unique name assigned to the structure.

dwId: A unique integer assigned to the structure.

qltType: The [QUEUELINK_TYPE](#) appropriate to the structure.

3 Protocol Details

3.1 Common Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The following information model is assumed by this protocol.

The smallest atomic object is a message. Messages may be individually viewed, deleted, or otherwise manipulated. Creation and alteration of messages is not supported.

Messages bound for the same destination (as determined by the domain part of the recipient addresses) are gathered into queues.

Two or more queues bound for ultimately different destinations may have the same next "hop" (relay destination). The server gathers into groups the queues with the same next hop in preparation for physical transfer. Each such group is called a "link".

Queues, links, and message transfers are managed by one or more virtual servers. The virtual servers are independent, named entities within the physical server. A single physical server contains one or more virtual servers.

Thus, the hierarchy of objects is accessible through a succession of interfaces is listed as follows. Each interface serves as a container and provides a method for listing and finding its contained objects.

- A physical server contains one or more virtual servers.
- A virtual server contains zero or more links.
- A link contains one or more queues.
- A queue contains zero or more messages.
- A message has content and other attributes.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Message Processing Events and Sequencing Rules

None.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

3.2 IAQAdmin

The **IAQAdmin** interface provides one method for a client to access Advanced Queue admin objects.

This interface inherits from IUnknown, as specified in [\[MS-DCOM\]](#) section 3.1.1.5.8. The version for this interface is 0.0.

The server MUST implement a DCOM Remote Protocol object class with the **class identifier (CLSID)** CLSID_AQAdmin ({83866cad-740d-11d2-94e4-00c04fa379f1}) using the **UUID** {476D70A6-1A90-11d3-BFCB-00C04FA3490A } for this interface. This interface includes the following methods beyond those of IUnknown.

Methods in RPC Opnum Order

Method	Description
GetVirtualServerAdminITF	Gets an administrative interface to an SMTP or message transfer agent (MTA) virtual server's queues. Opnum: 3

All methods MUST NOT throw exceptions.

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Message Processing Events and Sequence Rules

3.2.4.1 GetVirtualServerAdminITF (Opnum 3)

The **GetVirtualServerAdminITF** method is called by a client to get an administrative interface to an SMTP or MTA virtual server's queues.

```
HRESULT GetVirtualServerAdminITF(  
    [in] LPCWSTR wszComputer,  
    [in] LPCWSTR wszVirtualServer,  
    [out] IVSAQAdmin** ppivsaqadmin
```

);

wszComputer: A null-terminated Unicode (UCS-2) string that is the hostname of the physical computer where the virtual server is running. If this is NULL, the call is directed to a local physical computer.

wszVirtualServer: A null-terminated Unicode (UCS-2) numeric string identifying the requested virtual server. It MUST NOT be NULL.

ppivsaqadmin: The value assigned to the address of the requested virtual server advanced queue admin interface.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.
0x80070057 E_INVALIDARG	Invalid parameter.
0x80007005 E_ACCESSDENIED	Logged-on principal is not authorized to view queues on the specified virtual server.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to return a valid [IVSAQAdmin](#) interface pointer to the client, and return S_OK upon success. Otherwise, this method MUST return an appropriate error code.

3.2.5 Timer Events

None.

3.2.6 Other Local Events

None.

3.3 IVSAQAdmin

The **IVSAQAdmin** interface provides methods to access the list of links on a virtual server.

This interface inherits from IUnknown, as specified in [\[MS-DCOM\]](#) section 3.1.1.5.8. The UUID of the RPC interface for **IVSAQAdmin** is {e2ed3340-1e96-11d3-bfcc-00c04fa3490a}. The version for this interface is 0.0.

This interface includes the following methods beyond those of IUnknown.

Methods in RPC Opnum Order

Method	Description
GetLinkEnum	Gets an enumerator for all the links on this virtual server. Opnum: 3
StopAllLinks	Stops activities on the links of a virtual server. Opnum: 4
StartAllLinks	Starts all activities on the links of a virtual server. Opnum: 5
GetGlobalLinkState	Checks the global state of the links as set by StopAllLinks/StartAllLinks. Opnum: 6

All methods MUST NOT throw exceptions.

3.3.1 Abstract Data Model

None.

3.3.2 Timers

None.

3.3.3 Initialization

None.

3.3.4 Message Processing Events and Sequencing Rules

3.3.4.1 GetLinkEnum (Opnum 3)

The **GetLinkEnum** method is called by a client to get an enumerator to all the links within the virtual server.

```
HRESULT GetLinkEnum(  
    [out] IEnumVSAQLinks** ppenum  
);
```

ppenum: Upon success, this pointer MUST be the valid address of an [IEnumVSAQLinks](#) interface.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000	Success.

Return value/code	Description
S_OK	
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.
0x80070057 E_INVALIDARG	Invalid parameter.
0x80007005 E_ACCESSDENIED	Logged-on principal is not authorized to view queues on the specified virtual server.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to return a valid **IEnumVSAQLinks** enumerator interface to the client. Upon success, the method returns S_OK. Otherwise, it returns the appropriate error code.

3.3.4.2 StopAllLinks (Opnum 4)

The **StopAllLinks** method is called by a client to stop all activities on the links of a virtual server.

```
HRESULT StopAllLinks();
```

This method has no parameters.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to stop all the links in the virtual server. Upon success, the method returns S_OK. Otherwise, it returns the appropriate error code.

3.3.4.3 StartAllLinks (Opnum 5)

The **StartAllLinks** method is called by a client to start all activities on the links of a virtual server.

```
HRESULT StartAllLinks();
```

This method has no parameters.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to start all the links in the virtual server. Upon success, the method returns S_OK. Otherwise, it returns the appropriate error code.

3.3.4.4 GetGlobalLinkState (Opnum 6)

The **GetGlobalLinkState** method is called by a client to check the global state of the links as set by [StopAllLinks](#) or [StartAllLinks](#).

```
HRESULT GetGlobalLinkState();
```

This method has no parameters.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x00000001 S_FALSE	Links have been stopped (by a previous call to StopAllLinks).
0x80004001 E_NOTIMPL	Method not implemented.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to return the global state of the links, which has been set by the **StartAllLinks** and **StopAllLinks** functions. Upon success, the method returns S_OK. Otherwise, it returns the appropriate error code.

3.3.5 Timer Events

None.

3.3.6 Other Local Events

None.

3.4 IEnumVSAQLinks

The **IEnumVSAQLinks** interface provides access to individual links or groups of links through a **cursor** acquired through the [IVSAQAdmin](#) interface.

This interface inherits from IUnknown, as specified in [\[MS-DCOM\]](#) section 3.1.1.5.8. The UUID of the RPC interface for **IEnumVSAQLinks** is {ba7af300-7373-11d2-94e4-00c04fa379f1}. The version for this interface is 0.0.

This interface includes the following methods beyond those of IUnknown.

Methods in RPC Opnum Order

Method	Description
Next	Gets the requested number of links following the cursor position in the list of links. Opnum: 3
Skip	Moves the cursor through the list of IVSAQLink links. Opnum: 4
Reset	Resets the cursor to the first position in the list of IVSAQLink links. Opnum: 5
Clone	Creates a copy of the IEnumVSAQLinks interface. Opnum: 6

All methods MUST NOT throw exceptions.

3.4.1 Abstract Data Model

None.

3.4.2 Timers

None.

3.4.3 Initialization

None.

3.4.4 Message Processing Events and Sequencing Rules

3.4.4.1 Next (Opnum 3)

The **Next** method is called by a client to return the requested number of links following the cursor position in the list of links.

```
HRESULT Next(  
    [in] ULONG cElt,  
    [out] IVSAQLink** rgelt,  
    [out] ULONG* pcFetched  
);
```

cElt: The number of [IVSAQLink](#) references requested.

rgelt: The array in which the requested **IVSAQLink** references will be returned. The array MUST be at least as large as the number of objects requested.

pcFetched: The number of references actually returned.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	The requested number of references was retrieved.
0x00000001 S_FALSE	Fewer than the requested number of references was retrieved.

When this method is invoked, the server MUST attempt to return an array of valid **IVSAQLink** interface pointers less than or equal to the number of references requested, indicating the successful number. If less than the number requested, the server must indicate the difference with S_FALSE.

3.4.4.2 Skip (Opnum 4)

The **Skip** method is called by a client to move the cursor through the list of [IVSAQLink](#) links.

```
HRESULT Skip(  
    [in] ULONG cElt  
);
```

cElt: The number of **IVSAQLink** references through which the cursor is to be moved.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	The cursor was moved the requested number of positions.
0x00000001 S_FALSE	The list contained an insufficient number of links to move the cursor the requested number of positions. The cursor will be left at the last link in the list.

When this method is invoked, the server MUST attempt to move the cursor through the list the requested number of **IVSAQLink** references. If the list contains an insufficient number of links to accomplish the requested move, the server MUST set the cursor at the last link in the list.

3.4.4.3 Reset (Opnum 5)

The **Reset** method is called by a client to reset the cursor to the first position in the list of [IVSAQLink](#) links.

```
HRESULT Reset();
```

This method has no parameters.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	The cursor was reset to the beginning of the list.

When this method is invoked, the server MUST attempt to reset the cursor position at the beginning of the list.

3.4.4.4 Clone (Opnum 6)

The **Clone** method is called by a client to return a copy of the [IEnumVSAQLinks](#) interface.

```
HRESULT Clone(  
    [out] IEnumVSAQLinks** ppenum  
);
```

ppenum: The address of a reference to **IEnumVSAQLinks** into which the reference to the cloned interface will be loaded.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.

When this method is invoked, the server MUST attempt to create a copy of the **IEnumVSAQLinks** interface enumerator. On failure, the server MUST return the appropriate error code.

3.4.5 Timer Events

None.

3.4.6 Other Local Events

None.

3.5 IVSAQLink

The **IVSAQLink** interface provides access to a single virtual server AQ Link. A link represents a connectable entity. Another way to think of a link is that it represents the next hop to which mail messages on this link will be delivered.

This interface inherits from IUnknown, as specified in [\[MS-DCOM\]](#). The UUID of the RPC interface for the **IVSAQLink** is {3f962f94-1ecd-11d3-bfcc-00c04fa3490a}. The version for this interface is 0.0.

This interface includes the following methods beyond those of IUnknown.

Methods in RPC Opnum Order

Method	Description
GetInfo	Returns information about the link. Opnum: 3
SetLinkState	Changes the state of a link. Opnum: 4
GetQueueEnum	Provides an enumerator object for iterating through the queues associated with the link. Opnum: 5

All methods MUST NOT throw exceptions.

3.5.1 Abstract Data Model

None.

3.5.2 Timers

None.

3.5.3 Initialization

None.

3.5.4 Message Processing Events and Sequencing Rules

None.

3.5.4.1 GetInfo (Opnum 3)

The **GetInfo** method is called by a client to get information about the link.

```
HRESULT GetInfo(  
    [in, out] PLINK_INFO pli  
);
```

pli: MUST be a non-NULL pointer to the [LINK_INFO](#) structure with valid values as specified in section [2.2.9](#).

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.
0x80070057 E_INVALIDARG	Invalid parameter.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to fill the **LINK_INFO** structure with the link information. On failure, the server MUST return the appropriate error code.

3.5.4.2 SetLinkState (Opnum 4)

The **SetLinkState** method is called by a client to change the state of the link.

```
HRESULT SetLinkState(
    [in] LINK_ACTION la
);
```

la: Specifies the action to apply to the link. It MUST be one of the values defined in [LINK_ACTION](#).

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.
0x80070057 E_INVALIDARG	Invalid parameter.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to accomplish the action requested by the *la* parameter. On failure, the server MUST return the appropriate error code.

3.5.4.3 GetQueueEnum (Opnum 5)

The **GetQueueEnum** method is called by a client to get an enumerator object for iterating through the queues associated with the link.

```
HRESULT GetQueueEnum(  
    [out] IEnumLinkQueues** ppenum  
);
```

ppenum: MUST be the valid address of a [IEnumLinkQueues](#) pointer.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.
0x80070057 E_INVALIDARG	Invalid parameter.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to return a valid **IEnumLinkQueues** enumerator interface. On failure, the server MUST return the appropriate error code.

3.5.5 Timer Events

None.

3.5.6 Other Local Events

None.

3.6 IEnumLinkQueues

The **IEnumLinkQueues** interface provides access to individual links and groups of links.

This interface inherits from IUnknown, as specified in [\[MS-DCOM\]](#) section 3.1.1.5.8. The UUID of the RPC interface for the **IEnumLinkQueues** is {ba7af303-7373-11d2-94e4-00c04fa379f1}. The version for this interface is 0.0.

This interface includes the following methods beyond those of IUnknown.

Methods in RPC Opnum Order

Method	Description
Next	Gets the sequentially consecutive specified number of ILinkQueue objects. Opnum: 3
Skip	Moves the cursor through the list of ILinkQueue references. Opnum: 4
Reset	Resets the cursor to the first position in the list of ILinkQueue references. Opnum: 5
Clone	Creates a copy of the IEnumLinkQueues interface. Opnum: 6

All methods MUST NOT throw exceptions.

3.6.1 Abstract Data Model

None.

3.6.2 Timers

None.

3.6.3 Initialization

None.

3.6.4 Message Processing Events and Sequencing Rules

3.6.4.1 Next (Opnum 3)

The **Next** method is called by a client to get the sequentially consecutive specified number of [ILinkQueue](#) objects.

```
HRESULT Next(  
    [in] ULONG cElt,  
    [out] ILinkQueue** rgelt,  
    [out] ULONG* pcFetched  
);
```

cElt: The number of **ILinkQueue** references requested.

rgelt: The array where the requested **ILinkQueue** references will be returned. The array MUST be at least as large as the number of references requested.

pcFetched: The number of references returned.

Return Values: This method MUST return S_OK (0x00000000) on success and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	The requested number of references was retrieved.
0x00000001 S_FALSE	A number smaller than the requested number of references was retrieved.

When this method is invoked, the server MUST attempt to return an array of **ILinkQueue** interface pointers equal to or less than the number requested. Otherwise, the server will return the number of available interface pointers.

3.6.4.2 Skip (Opnum 4)

The **Skip** method is called by a client to move a cursor through the list of [ILinkQueue](#) references.

```
HRESULT Skip(
    [in] ULONG cElt
);
```

cElt: The number of **ILinkQueue** references through which the cursor is to be moved.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	The cursor was moved the requested number of positions.
0x00000001 S_FALSE	The list contained an insufficient number of objects. The cursor will be left at the last object in the list.

When this method is invoked, the server MUST attempt to move the cursor through the list the specified number of **ILinkQueue** references requested. If the available number of references in the list is less than the number requested, the server MUST set the cursor at the last in the list.

3.6.4.3 Reset (Opnum 5)

The **Reset** method is called by a client to reset the cursor to the first position in the list of [ILinkQueue](#) references.

```
HRESULT Reset();
```

This method has no parameters.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies results specific to this method.

Return value/code	Description
0x00000000 S_OK	The cursor was reset to the beginning of the list.

When this method is invoked, the server MUST attempt to reset the cursor to the first position in the list of **ILinkQueue** references.

3.6.4.4 Clone (Opnum 6)

The **Clone** method is called by a client to get a copy of the [IEnumLinkQueues](#) interface.

```
HRESULT Clone(
    [out] IEnumLinkQueues** ppenum
);
```

ppenum: The address of a reference to **IEnumLinkQueues** into which the reference to the cloned interface will be loaded.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.

When this method is invoked, the server MUST attempt to return the copy of the **IEnumLinkQueues** enumerator.

3.6.5 Timer Events

None.

3.6.6 Other Local Events

None.

3.7 ILinkQueue

The **ILinkQueue** interface provides access to individual queues in the link. A queue contains a list of messages bound for the same destination. All messages in a queue have the same SMTP domain part.

This interface inherits from IUnknown, as specified in [\[MS-DCOM\]](#) section 3.1.1.5.8. The UUID of the RPC interface for the **ILinkQueue** is {ff9a1bb6-1e96-11d3-bfcc-00c04fa3490a}. The version for this interface is 0.0.

This interface includes the following methods beyond those of IUnknown.

Methods in RPC Opnum Order

Method	Description
GetInfo	Returns information about the queue. Opnum: 3
GetMessageEnum	Provides an enumerator object for iterating through the messages associated with the queue. Opnum: 4

All methods MUST NOT throw exceptions.

3.7.1 Abstract Data Model

None.

3.7.2 Timers

None.

3.7.3 Initialization

None.

3.7.4 Message Processing Events and Sequencing Rules

3.7.4.1 GetInfo (Opnum 3)

The **GetInfo** method is called by a client to get information about the queue.

```
HRESULT GetInfo(  
    [in, out] PQUEUE_INFO pqi  
);
```

pqi: MUST be a non-NULL pointer to the [QUEUE_INFO](#) structure with valid values as specified in section [2.2.11](#).

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.

Return value/code	Description
0x80070057 E_INVALIDARG	Invalid parameter.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to fill the user's **QUEUE_INFO** structure with valid values.

3.7.4.2 GetMessageEnum (Opnum 4)

The **GetMessageEnum** method is called by a client to get an enumerator object for iterating through the messages associated with the queue.

```
HRESULT GetMessageEnum(
    [in] PMESSAGE_ENUM_FILTER pFilter,
    [out] IAQEnumMessages** ppenum
);
```

pFilter: MUST be a valid filter specification as defined in [MESSAGE_ENUM_FILTER](#).

ppenum: MUST be the valid address of a [IAQEnumMessages](#) pointer.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [MS-ERREF](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.
0x80070057 E_INVALIDARG	Invalid parameter.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to return a message enumerator interface **IAQEnumMessages**.

3.7.5 Timer Events

None.

3.7.6 Other Local Events

None.

3.8 IAQEnumMessages

The **IAQEnumMessages** interface provides access to individual messages and groups of messages in a queue.

This interface inherits from IUnknown, as specified in [\[MS-DCOM\]](#) section 3.1.1.5.8. The UUID of the RPC interface for **IAQEnumMessages** is {ba7af302-7373-11d2-94e4-00c04fa379f1}. The version for this interface is 0.0.

This interface includes the following methods beyond those of IUnknown.

Methods in RPC Opnum Order

Method	Description
Next	Gets the sequentially consecutive specified number of IAQMessage objects. Opnum: 3
Skip	Moves the cursor through the list of IAQMessage references. Opnum: 4
Reset	Resets the cursor to the first position in the list of IAQMessage references. Opnum: 5
Clone	Creates a copy of this enumerator object. Opnum: 6

All methods MUST NOT throw exceptions.

3.8.1 Abstract Data Model

None.

3.8.2 Timers

None.

3.8.3 Initialization

None.

3.8.4 Message Processing Events and Sequencing Rules

3.8.4.1 Next (Opnum 3)

The **Next** method is called by a client to get the sequentially consecutive specified number of [IAQMessage](#) objects.

```
HRESULT Next(  
    [in] ULONG cElt,
```

```

[out] IAQMessage** rgelt,
[out] ULONG* pcFetched
);

```

cElt: The number of **IAQMessage** references requested.

rgelt: The array in which the requested **IAQMessage** references will be returned. The array MUST be at least as large as the number of references requested.

pcFetched: The number of references actually returned.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	The requested number of references was retrieved.
0x00000001 S_FALSE	Fewer than the requested number of references was retrieved.

When this method is invoked, the server MUST attempt to return an array of **IAQMessage** interface pointers less than or equal to the number requested. The returned array will have the specified number of interface pointers. Otherwise, the server will return the maximum number available with the error code S_FALSE.

3.8.4.2 Skip (Opnum 4)

The **Skip** method is called by a client to move the cursor through the list of [IAQMessage](#) references.

```

HRESULT Skip(
[in] ULONG cElt
);

```

cElt: The number of **IAQMessage** references through which the cursor is to be moved.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	The cursor was moved the requested number of positions.
0x00000001 S_FALSE	The list contained an insufficient number of objects. The cursor will be positioned at the last reference in the list.

When this method is invoked, the server MUST attempt to move the cursor the specified number of **IAQMessage** references in the list. If the number requested is greater than the number available, the cursor MUST be moved to the end of the list, and S_FALSE returned to the user.

3.8.4.3 Reset (Opnum 5)

The **Reset** method is called by a client to reset to the first position in the list of [IAQMessage](#) references.

```
HRESULT Reset();
```

This method has no parameters.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies results specific to this method.

Return value/code	Description
0x00000000 S_OK	The cursor was reset to the beginning of the list.

When this method is invoked, the server MUST reset the cursor to the first position in the list.

3.8.4.4 Clone (Opnum 6)

The **Clone** method is called by a client to create a clone of the enumerator object.

```
HRESULT Clone(
    [out] IAQEnumMessages** ppenum
);
```

ppenum: The address of a reference to [IAQEnumMessages](#) into which the reference to the cloned interface will be loaded.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.

When this method is invoked, the server MUST attempt to return a copy of the **IAQEnumMessages** enumerator. Otherwise, the server MUST return the appropriate error code.

3.8.5 Timer Events

None.

3.8.6 Other Local Events

None.

3.9 IAQMessage

The **IAQMessage** interface provides methods for a client to access a single message.

This interface inherits from IUnknown, as specified in [\[MS-DCOM\]](#) section 3.1.1.5.8. The UUID of the RPC interface for **IAQMessage** is {ba7af305-7373-11d2-94e4-00c04fa379f1}. The version for this interface is 0.0.

This interface includes the following methods beyond those of IUnknown.

Methods in RPC Opnum Order

Method	Description
GetInfo	Gets information about the queue. Opnum: 3
GetContentStream	Gets a stream for the message content. Opnum: 4

All methods MUST NOT throw exceptions.

3.9.1 Abstract Data Model

None.

3.9.2 Timers

None.

3.9.3 Initialization

None.

3.9.4 Message Processing Events and Sequencing Rules

None.

3.9.4.1 GetInfo (Opnum 3)

The **GetInfo** method is called by a client to get information about a message.

```
HRESULT GetInfo(  
    [in, out] PMESSAGE_INFO pmi  
);
```


pmi: Must be a non-NULL pointer to the [MESSAGE_INFO](#) structure with valid values as specified in section [2.2.12](#).

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x80004001 E_NOTIMPL	Method not implemented.
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.
0x80070057 E_INVALIDARG	Invalid parameter.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to return a valid pointer to the **MESSAGE_INFO** structure. Otherwise, the server MUST return the appropriate error code.

3.9.4.2 GetContentStream (Opnum 4)

The **GetContentStream** method is called by a client to get the content of a message.

```
HRESULT GetContentStream(  
    [out] IStream** ppIStream,  
    [out] LPWSTR* pwszContentType  
);
```

ppIStream: MUST be the address of an actual **IStream** pointer. **IStream** is a standard COM interface. For more information, see [\[MSDN-COM\]](#).

pwszContentType: MUST be the address of an actual Unicode (UCS-2) string pointer that will return a string describing the content.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x80004001	Method not implemented.

Return value/code	Description
E_NOTIMPL	
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.
0x80070057 E_INVALIDARG	Invalid parameter.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to return a valid pointer to the **IStream** interface to retrieve message content.

3.9.5 Timer Events

None.

3.9.6 Other Local Events

None.

3.10 IAQMessageAction

The **IAQMessageAction** interface is used to apply specified actions to messages on this virtual server that match the specified message filter criteria. The actions will be applied only to messages that belong to the object implementing this interface. For example, only messages that match the filter on a given queue will be affected if this method is called on an [ILinkQueue](#) object.

This interface inherits from IUnknown, as specified in [\[MS-DCOM\]](#) section 3.1.1.5.8. The UUID of the RPC interface for **IAQMessageAction** is {1eb44a71-1e95-11d3-bfcc-00c04fa3490a}. The version for this interface is 0.0.

This interface includes the following methods beyond those of IUnknown.

Methods in RPC Opnum Order

Method	Description
ApplyActionToMessages	Apply the specified action to messages on this virtual server that match the specified message filter criteria. Opnum: 3
QuerySupportedActions	Describes which actions/filters of the ApplyActionToMessages method are supported. Opnum: 4

All methods MUST NOT throw exceptions.

3.10.1 Abstract Data Model

None.

3.10.2 Timers

None.

3.10.3 Initialization

None.

3.10.4 Message Processing Events and Sequencing Rules

3.10.4.1 ApplyActionToMessages (Opnum 3)

The **ApplyActionToMessages** method is called by a client to apply specified actions to messages on this virtual server that match the specified message filter criteria. The actions will only be applied to messages that belong to the object that implements this interface.

For example, only messages that match the filter on a given queue will be affected if this method is called on an [ILinkQueue](#) object.

```
HRESULT ApplyActionToMessages (  
    [in] PMESSAGE_FILTER Filter,  
    [in] MESSAGE_ACTION Action,  
    [out] DWORD* pcMsgs  
);
```

Filter: The criteria for selection of messages formatted according to the [MESSAGE_FILTER](#) structure definition. The argument MUST be a non-NULL pointer to such a structure. The filter MAY specify formatting that does not match existing messages.

Action: The action to be applied to the messages selected by the filter. The argument MUST be a value defined by the [MESSAGE_ACTION](#) enumeration.

pcMsgs: This argument MUST be a non-NULL pointer to a [DWORD](#). On execution, the method returns via this argument the number of messages to which the action was applied.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.
0x80070057	Invalid parameter.

Return value/code	Description
E_INVALIDARG	
0x80007005 E_ACCESSDENIED	Logged-on principal is not authorized to view queues on the specified virtual server.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to return a **DWORD** value indicating the number of messages to which the action was applied. Upon method or action failure, the server MUST return the appropriate error code.

3.10.4.2 QuerySupportedActions (Opnum 4)

The **QuerySupportedActions** method is called by a client to discover the list of supported actions and filters.

```
HRESULT QuerySupportedActions (
    [out] DWORD* pdwSupportedActions,
    [out] DWORD* pdwSupportedFilterFlags
);
```

pdwSupportedActions: MUST be a non-NULL pointer to a **DWORD** that returns a bitfield indicating the supported actions from values specified in [MESSAGE ACTION](#).

pdwSupportedFilterFlags: MUST be a non-NULL pointer to a **DWORD** that returns a bitfield indicating the supported filters from values specified in [MESSAGE FILTER FLAGS](#).

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x8007000E E_OUTOFMEMORY	Insufficient resources.
0x80000005 E_POINTER	NULL pointer parameter.
0x80070057 E_INVALIDARG	Invalid parameter.
0x80007005 E_ACCESSDENIED	Logged-on principal is not authorized to view queues on the specified virtual server.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to return a bitfield indicating the supported filters from values defined in **MESSAGE_FILTER_FLAGS**. On failure, the server MUST return the appropriate error code.

3.10.5 Timer Events

None.

3.10.6 Other Local Events

None.

3.11 IUniqueId

The **IUniqueId** interface provides methods for a client to get the canonical name of a queue or link.

This interface inherits from IUnknown, as specified in [\[MS-DCOM\]](#) section 3.1.1.5.8. The UUID of the RPC interface for **IUniqueId** is {EA4DFDF2-9E87-4c57-B845-123872C5649F}. The version for this interface is 0.0.

This interface includes the following methods beyond those of IUnknown.

Methods in RPC Opnum Order

Method	Description
GetUniqueId	Return QUEUELINK_ID for the queue or link. Opnum: 3

All methods MUST NOT throw exceptions.

3.11.1 Abstract Data Model

None.

3.11.2 Timers

None.

3.11.3 Initialization

None.

3.11.4 Message Processing Events and Sequencing Rules

3.11.4.1 GetUniqueId (Opnum 3)

The **GetUniqueId** method is called by a client to get a [QUEUELINK_ID](#) for this queue or link. This is the canonical representation of the queue. The queue API guarantees that there is no more than one queue or link with the same **QUEUELINK_ID** at a given point in time. The memory associated with the out parameter is guaranteed good until the underlying queue API object is released.

```
HRESULT GetUniqueId(  
    [out] QUEUELINK_ID** ppqlid
```

);

ppqlid: MUST be a valid address to the **QUEUELINK_ID** pointer.

Return Values: This method MUST return S_OK (0x00000000) on success, and a failure result (as specified in [\[MS-ERREF\]](#) section 2) on failure. All failure results MUST be treated identically. The following table specifies failure results specific to this method.

Return value/code	Description
0x00000000 S_OK	Success.
0x80000005 E_POINTER	NULL pointer parameter.
0x80004001 E_NOTIMPL	Method not implemented.
0x80071722 RPC_S_SERVER_UNAVAILABLE	Unable to connect to the specified virtual server. Returned as the result of HRESULT_FROM_WIN32.

When this method is invoked, the server MUST attempt to return a valid pointer to a **QUEUELINK_ID** structure. Otherwise, the server MUST return the appropriate error code.

3.11.5 Timer Events

None.

3.11.6 Other Local Events

None.

4 Protocol Examples

The hierarchy of objects is accessible through a succession of interfaces.

The [IAQAdmin](#) interface provides a method called [GetVirtualServerAdminITF](#) for a server to send the client an interface ([IVSAQAdmin](#)) to the SMTP virtual server advance queue.

The **IVSAQAdmin** interface provides a method called [GetLinkEnum](#) for a server to send an enumerator interface ([IEnumVSAQLinks](#)) that can be used to get all the links in this virtual server.

This interface also provides a method called [StartAllLinks](#) for a server to start all the links in the SMTP virtual server.

This interface also provides a method called [StopAllLinks](#) for a server to stop all the links in the SMTP virtual server.

This interface also provides a method called [GetGlobalLinkState](#) for a server to return the global state of the links.

The **IEnumVSAQLinks** interface that is returned provides a method to enumerate references to all the links in the virtual server. References to the links will be returned through the [IVSAQLink](#) interface.

The **IVSAQLink** interface provides a method called [GetInfo](#) to get information about each link.

This **IVSAQLink** interface also provides a method called [SetLinkState](#) to set the state of each link.

This **IVSAQLink** interface also provides a method called [GetQueueEnum](#) to retrieve an enumerator interface ([IEnumLinkQueues](#)) to queues in a link.

The **IEnumLinkQueues** interface provides methods to enumerate or retrieve queues in a link. The reference to the queue in this link will be returned through the [ILinkQueue](#) interface.

The **ILinkQueue** interface provides methods to get information about queues and the message enumerator interface ([IAQEnumMessages](#)).

The **IAQEnumMessages** interface provides methods to iterate over messages in a queue. This interface also provides a reference to a message through the [IAQMessage](#) interface.

The **IAQMessage** interface provides methods to retrieve information about messages. This interface also provides a method called [GetContentStream](#) to retrieve the contents of a message through an **IStream** pointer (see [\[MSDN-COM\]](#)).

The following example shows a client application for getting a reference to the virtual server advance queue admin interface (**IVSAQAdmin**), and starting all the links in the virtual server. This example also shows how to enumerate all the links in the virtual server.

```
int __cdecl main(int argc, char **argv)
{
    HRESULT hr = S_OK;
    IAQAdmin *pIAQAdmin = NULL;
    IVSAQAdmin *pIVSAQAdmin = NULL;
    IEnumVSAQLinks *pIEnumVSAQLinks = NULL;
    IVSAQLink *pILink = NULL;

    // Initialize COM
    hr = CoInitializeEx(NULL, COINIT_MULTITHREADED);
```

```

if(FAILED(hr))
{
    printf("CoInitializeEx failed w/ 0x%x\n", hr);
    return hr;
}

// Create Advance Queue Administration class
hr = CoCreateInstance( CLSID_AQAdmin,
                      NULL,
                      CLSCTX_INPROC_SERVER,
                      IID_IAQAdmin,
                      (void **) &pIAQAdmin);

if(FAILED(hr))
{
    printf("CoCreateInstance failed with 0x%x\n", hr);
    goto Exit;
}

// Get the admin interface to first Virtual Server
hr = m_pIAQAdmin->GetVirtualServerAdminITF(NULL,
                                           L"1",
                                           &pIVSAQAdmin);

if(FAILED(hr))
{
    printf("GetVirtualServerAdminITF failed with 0x%x\n", hr);
    goto Exit;
}

// start all the link in this virtual server.
hr = pIVSAQAdmin->StartAllLinks();
if(FAILED(hr))
{
    printf("StartAllLinks failed with 0x%x\n", hr);
    goto Exit;
}

printf("StartAllLinks succeeded\n", hr);

// Now get a reference to IEnumVSAQLinks enumerator interface
// to iterate all the links in this virtual server.
hr = pIVSAQAdmin->GetLinkEnum(&pIEnumVSAQLinks);
if(FAILED(hr))
{
    printf("GetLinkEnum failed with 0x%x\n", hr);
    goto Exit;
}

LINK_INFO linkInfo;
ZeroMemory(&linkInfo, sizeof(LINK_INFO));

for(int nCrtLink = 1; TRUE; nCrtLink++)
{
    // get the pointer to the link interface
    hr = GetLink(pIEnumVSAQLinks, &pILink, &linkInfo);
    if(hr == S_FALSE)
    {
        if(nCrtLink == 1)
        {
            puts("No links.");
        }
    }
}

```



```

        break;
    }
    else if(FAILED(hr))
    {
        break;
    }
    else if(hr == S_OK)
    {
        // Succeeded.
        // Enumerate queues in the link similar way
    }

    if(NULL != pLink)
    {
        pLink->Release();
        pLink = NULL;
    }
    FreeStruct(&linkInfo);
}

Exit:
if (NULL != pIVSAQAdmin) pIVSAQAdmin->Release();
if (NULL != pIVSAQAdmin) pIVSAQAdmin->Release();
if (NULL != pIEnumVSAQLinks) pIEnumVSAQLinks->Release();
if(NULL != pLink) pLink->Release();
}

HRESULT GetLink( IN IEnumVSAQLinks *pIEnumVSAQLinks,
                OUT IVSAQLink **ppILink,
                IN OUT LINK_INFO *pLinkInfo)
{
    HRESULT hr;
    DWORD cFetched;

    hr = pIEnumVSAQLinks->Next(1, ppILink, &cFetched);
    if(hr == S_FALSE)
    {
        goto Exit;
    }
    else if(FAILED(hr))
    {
        printf("pIEnumVSAQLinks->Next failed with 0x%x\n", hr);
        goto Exit;
    }
    else if(NULL == (*ppILink))
    {
        printf("pLink is NULL.\n", hr);
        goto Exit;
    }
    else
    {
        ZeroMemory(pLinkInfo, sizeof(LINK_INFO));
        pLinkInfo->dwVersion = CURRENT_QUEUE_ADMIN_VERSION;
        hr = (*ppILink)->GetInfo(pLinkInfo);
        if(FAILED(hr))
        {
            printf("pLink->GetInfo failed with 0x%x\n", hr);
            if ( HRESULT_FROM_WIN32(RPC_S_SERVER_UNAVAILABLE)
                == hr )

```

```

        printf("RPC Server Unavailable.\n");
    else if ( hr == E_POINTER )
        printf("Null pointer.\n");
    else if ( hr == E_OUTOFMEMORY )
        printf("Out of memory.\n");
    else if ( hr == E_INVALIDARG )
        printf("Invalid argument.\n");
    else
        printf("Unknown error.\n");
    goto Exit;
    }
}

Exit:
    return hr;
}

void FreeStruct(LINK_INFO *pStruct)
{
    if(NULL != pStruct->szLinkName)
    {
        pStruct->szLinkName = NULL;
    }
}

```

5 Security

5.1 Security Considerations for Implementers

Implementers should review the security considerations listed in [\[MS-DCOM\]](#) section 5.1, as these are also valid for the IIS IAQ AdminRPC Protocol.

5.2 Index of Security Parameters

None.

6 Appendix A: Full IDL

For ease of implementation, the full **IDL** is provided, where "ms-dtyp.idl" is the IDL specified in [\[MS-DTYP\] Appendix A](#).

```
import "ms-dtyp.idl";
import "ms-oaut.idl";

[
    object,
    uuid(0c733a30-2a1c-11ce-ade5-00aa0044773d),
    pointer_default(unique)
]
interface ISequentialStream : IUnknown
{
    [local]
    HRESULT Read(
        [out, size_is(cb), length_is(*pcbRead)]
        void *pv,
        [in] ULONG cb,
        [out] ULONG *pcbRead);

    [call_as(Read)]
    HRESULT RemoteRead(
        [out, size_is(cb), length_is(*pcbRead)]
        byte *pv,
        [in] ULONG cb,
        [out] ULONG *pcbRead);

    [local]
    HRESULT Write(
        [in, size_is(cb)] void const *pv,
        [in] ULONG cb,
        [out] ULONG *pcbWritten);

    [call_as(Write)]
    HRESULT RemoteWrite(
        [in, size_is(cb)] byte const *pv,
        [in] ULONG cb,
        [out] ULONG *pcbWritten);
}

[
    object,
    uuid(0000000c-0000-0000-C000-000000000046),
    pointer_default(unique)
]
interface IStream : ISequentialStream
{
    typedef [unique] IStream *LPSTREAM;

    /* Storage stat buffer */

    typedef struct tagSTATSTG
    {
        LPOLESTR pwcsName;
```

```

    DWORD type;
    ULARGE_INTEGER cbSize;
    FILETIME mtime;
    FILETIME ctime;
    FILETIME atime;
    DWORD grfMode;
    DWORD grfLocksSupported;
    CLSID clsid;
    DWORD grfStateBits;
    DWORD reserved;
} STATSTG;

/* Storage element types */
typedef enum tagSTGTY
{
    STGTY_STORAGE    = 1,
    STGTY_STREAM     = 2,
    STGTY_LOCKBYTES  = 3,
    STGTY_PROPERTY   = 4
} STGTY;

typedef enum tagSTREAM_SEEK
{
    STREAM_SEEK_SET = 0,
    STREAM_SEEK_CUR = 1,
    STREAM_SEEK_END = 2
} STREAM_SEEK;

typedef enum tagLOCKTYPE
{
    LOCK_WRITE      = 1,
    LOCK_EXCLUSIVE  = 2,
    LOCK_ONLYONCE   = 4
} LOCKTYPE;

[local]
HRESULT Seek(
    [in] LARGE_INTEGER dlibMove,
    [in] DWORD dwOrigin,
    [out] ULARGE_INTEGER *plibNewPosition);

[call_as(Seek)]
HRESULT RemoteSeek(
    [in] LARGE_INTEGER dlibMove,
    [in] DWORD dwOrigin,
    [out] ULARGE_INTEGER *plibNewPosition);

HRESULT SetSize(
    [in] ULARGE_INTEGER libNewSize);

[local]
HRESULT CopyTo(
    [in, unique] IStream *pstm,
    [in] ULARGE_INTEGER cb,
    [out] ULARGE_INTEGER *pcbRead,
    [out] ULARGE_INTEGER *pcbWritten);

[call_as(CopyTo)]
HRESULT RemoteCopyTo(

```

```

        [in, unique] IStream *pstm,
        [in] ULARGE_INTEGER cb,
        [out] ULARGE_INTEGER *pcbRead,
        [out] ULARGE_INTEGER *pcbWritten);

HRESULT Commit(
    [in] DWORD grfCommitFlags);

HRESULT Revert();

HRESULT LockRegion(
    [in] ULARGE_INTEGER libOffset,
    [in] ULARGE_INTEGER cb,
    [in] DWORD dwLockType);

HRESULT UnlockRegion(
    [in] ULARGE_INTEGER libOffset,
    [in] ULARGE_INTEGER cb,
    [in] DWORD dwLockType);

HRESULT Stat(
    [out] STATSTG *pstatstg,
    [in] DWORD grfStatFlag);

HRESULT Clone(
    [out] IStream **ppstm);
}

```

```

interface IVSAQAdmin;
interface IEnumVSAQLinks;
interface IAQEnumMessages;
interface IEnumLinkQueues;
interface IAQMessage;
interface IVSAQLink;
interface ILinkQueue;

```

```

typedef enum tagQUEUE_ADMIN_VERSIONS
{
    CURRENT_QUEUE_ADMIN_VERSION = 4
} QUEUE_ADMIN_VERSIONS;

```

```

typedef struct tagMESSAGE_FILTER {
    DWORD dwVersion;
    DWORD fFlags;
    [string] LPCWSTR szMessageId;
    [string] LPCWSTR szMessageSender;
    [string] LPCWSTR szMessageRecipient;
    DWORD dwLargerThanSize;
    SYSTEMTIME stOlderThan;
} MESSAGE_FILTER,
*PMESSAGE_FILTER;

```

```

typedef enum tagMESSAGE_FILTER_FLAGS
{
    MF_MESSAGEID    = 0x00000001,
    MF_SENDER      = 0x00000002,
    MF_RECIPIENT   = 0x00000004,
    MF_SIZE        = 0x00000008,
}

```

```

MF_TIME          = 0x00000010,
MF_FROZEN        = 0x00000020,
MF_FAILED        = 0x00000100,
MF_ALL           = 0x40000000,
MF_INVERTSENSE  = 0x80000000
} MESSAGE_FILTER_FLAGS;

typedef enum tagMESSAGE_ACTION
{
    MA_THAW_GLOBAL      = 0x00000001,
    MA_COUNT            = 0x00000002,
    MA_FREEZE_GLOBAL   = 0x00000004,
    MA_DELETE          = 0x00000008,
    MA_DELETE_SILENT   = 0x00000010
} MESSAGE_ACTION;

typedef enum tagMESSAGE_ENUM_FILTER_TYPE
{
    MEF_FIRST_N_MESSAGES = 0x00000001,
    MEF_SENDER           = 0x00000002,
    MEF_RECIPIENT       = 0x00000004,
    MEF_LARGER_THAN     = 0x00000008,
    MEF_OLDER_THAN      = 0x00000010,
    MEF_FROZEN          = 0x00000020,
    MEF_N_LARGEST_MESSAGES = 0x00000040,
    MEF_N_OLDEST_MESSAGES = 0x00000080,
    MEF_FAILED          = 0x00000100,
    MEF_ALL             = 0x40000000,
    MEF_INVERTSENSE     = 0x80000000
} MESSAGE_ENUM_FILTER_TYPE;

typedef struct tagMESSAGE_ENUM_FILTER {
    DWORD dwVersion;
    DWORD mefType;
    DWORD cMessages;
    DWORD cbSize;
    DWORD cSkipMessages;
    SYSTEMTIME stDate;
    [string] LPCWSTR szMessageSender;
    [string] LPCWSTR szMessageRecipient;
} MESSAGE_ENUM_FILTER,
*PMESSAGE_ENUM_FILTER;

typedef enum tagLINK_INFO_FLAGS
{
    LI_ACTIVE          = 0x00000001,
    LI_READY           = 0x00000002,
    LI_RETRY           = 0x00000004,
    LI_SCHEDULED       = 0x00000008,
    LI_REMOTE          = 0x00000010,
    LI_FROZEN          = 0x00000020,
    LI_TYPE_REMOTE_DELIVERY = 0x00000100,
    LI_TYPE_LOCAL_DELIVERY = 0x00000200,
    LI_TYPE_PENDING_ROUTING = 0x00000400,
    LI_TYPE_PENDING_CAT = 0x00000800,
    LI_TYPE_CURRENTLY_UNREACHABLE = 0x00001000,
    LI_TYPE_DEFERRED_DELIVERY = 0x00002000,
    LI_TYPE_INTERNAL   = 0x00004000,
    LI_TYPE_PENDING_SUBMIT = 0x00008000
}

```

```

} LINK_INFO_FLAGS;

typedef enum tagLINK_ACTION
{
    LA_INTERNAL = 0x00000000,
    LA_KICK = 0x00000001,
    LA_FREEZE = 0x00000020,
    LA_THAW = 0x00000040
} LINK_ACTION;

typedef struct tagLINK_INFO
{
    DWORD dwVersion;
    [string] LPWSTR szLinkName;
    DWORD cMessages;
    DWORD fStateFlags;
    SYSTEMTIME stNextScheduledConnection;
    SYSTEMTIME stOldestMessage;
    ULARGE_INTEGER cbLinkVolume;
    [string] LPWSTR szLinkDN;
    [string] LPWSTR szExtendedStateInfo;
    DWORD dwSupportedLinkActions;
} LINK_INFO,
*PLINK_INFO;

typedef struct tagQUEUE_INFO {
    DWORD dwVersion;
    [string] LPWSTR szQueueName;
    [string] LPWSTR szLinkName;
    DWORD cMessages;
    ULARGE_INTEGER cbQueueVolume;
    DWORD dwMsgEnumFlagsSupported;
} QUEUE_INFO,
*PQUEUE_INFO;

typedef enum tagAQ_MESSAGE_FLAGS
{
    MP_HIGH = 0x00000001,
    MP_NORMAL = 0x00000002,
    MP_LOW = 0x00000004,
    MP_MSG_FROZEN = 0x00000008,
    MP_MSG_RETRY = 0x00000010,
    MP_MSG_CONTENT_AVAILABLE = 0x00000020
} AQ_MESSAGE_FLAGS;

typedef struct tagMESSAGE_INFO
{
    DWORD dwVersion;
    [string] LPWSTR szMessageId;
    [string] LPWSTR szSender;
    [string] LPWSTR szSubject;
    DWORD cRecipients;
    [string] LPWSTR szRecipients;
    DWORD cCCRecipients;
    [string] LPWSTR szCCRecipients;
    DWORD cBCCRecipients;
    [string] LPWSTR szBCCRecipients;
    DWORD fMsgFlags;
    DWORD cbMessageSize;
}

```



```

SYSTEMTIME stSubmission;
SYSTEMTIME stReceived;
SYSTEMTIME stExpiry;
DWORD cFailures;
DWORD cEnvRecipients;
DWORD cbEnvRecipients;
[size_is(cbEnvRecipients/sizeof(WCHAR))]
    WCHAR* mszEnvRecipients;
} MESSAGE_INFO,
*PMESSAGE_INFO;

typedef enum tagQUEUELINK_TYPE
{
    QLT_QUEUE,
    QLT_LINK,
    QLT_NONE
} QUEUELINK_TYPE;

typedef struct tagQUEUELINK_ID
{
    GUID uuid;
    [string] LPWSTR szName;
    DWORD dwId;
    QUEUELINK_TYPE qltType;
} QUEUELINK_ID;

[
    helpstring("Advanced Queue Administration Object"),
    object,
    pointer_default(unique),
    uuid(476D70A6-1A90-11d3-BFCB-00C04FA3490A)
]
interface IAQAdmin : IUnknown
{
    HRESULT GetVirtualServerAdminITF(
        [in] LPCWSTR wszComputer,
        [in] LPCWSTR wszVirtualServer,
        [out] IVSAQAdmin **ppvsaqadmin
    );
}

[
    helpstring("Advanced Queue Message Action Interface"),
    object,
    pointer_default(unique),
    uuid(1EB44A71-1E95-11d3-BFCC-00C04FA3490A)
]
interface IAQMessageAction : IUnknown
{
    HRESULT ApplyActionToMessages(
        [in] PMESSAGE_FILTER Filter,
        [in] MESSAGE_ACTION Action,
        [out] DWORD *pcMsgs
    );

    HRESULT QuerySupportedActions(
        [out] DWORD *pdwSupportedActions,
        [out] DWORD *pdwSupportedFilterFlags

```

```

    );
}
/*
@interface IVSAQAdmin | Interface to per-virtual-server AQ
Administration
@meth HRESULT | GetLinkEnum | Get an enumerator over all links on
this VS
@meth HRESULT | StopAllLinks | Stop all active outbound links
@meth HRESULT | StartAllLinks | Start all eligible outbound links
*/
[
    helpstring("Advanced Queue Administration Object"),
    object,
    pointer_default(unique),
    uuid(E2ED3340-1E96-11d3-BFCC-00C04FA3490A)
]
interface IVSAQAdmin : IUnknown
{
    HRESULT GetLinkEnum(
        [out] IEnumVSAQLinks **ppenum
    );

    HRESULT StopAllLinks();

    HRESULT StartAllLinks();

    HRESULT GetGlobalLinkState();
}

[
    helpstring("Advanced Queue Administration Object"),
    object,
    pointer_default(unique),
    uuid(ba7af300-7373-11d2-94e4-00c04fa379f1)
]
interface IEnumVSAQLinks : IUnknown
{
    HRESULT Next(
        [in] ULONG cElt,
        [out] IVSAQLink **rgelt,
        [out] ULONG *pcFetched
    );

    HRESULT Skip(
        [in] ULONG cElt
    );

    HRESULT Reset();

    HRESULT Clone(
        [out] IEnumVSAQLinks **ppenum
    );
}

[
    helpstring("Advanced Queue Administration Object"),
    object,

```

```

        pointer_default(unique),
        uuid(3F962F94-1ECD-11d3-BFCC-00C04FA3490A)
    ]
interface IVSAQLink : IUnknown
{
    HRESULT GetInfo(
        [in, out] PLINK_INFO pli
    );

    HRESULT SetLinkState(
        [in] LINK_ACTION la
    );

    HRESULT GetQueueEnum(
        [out] IEnumLinkQueues **ppenum
    );
}

[
    helpstring("Advanced Queue Administration Object"),
    object,
    pointer_default(unique),
    uuid(ba7af302-7373-11d2-94e4-00c04fa379f1)
]
interface IAQEnumMessages : IUnknown
{
    HRESULT Next(
        [in] ULONG cElt,
        [out] IAQMessage **rgelt,
        [out] ULONG *pcFetched
    );

    HRESULT Skip(
        [in] ULONG cElt
    );

    HRESULT Reset();

    HRESULT Clone(
        [out] IAQEnumMessages **ppenum
    );
}

[
    helpstring("Advanced Queue Administration Object"),
    object,
    pointer_default(unique),
    uuid(ba7af303-7373-11d2-94e4-00c04fa379f1)
]
interface IEnumLinkQueues : IUnknown
{
    HRESULT Next(
        [in] ULONG cElt,
        [out] ILinkQueue **rgelt,
        [out] ULONG *pcFetched
    );

    HRESULT Skip(

```

```

        [in] ULONG cElt
    );

    HRESULT Reset();

    HRESULT Clone(
        [out] IEnumLinkQueues **ppenum
    );
}

[
    helpstring("Advanced Queue Administration Object"),
    object,
    pointer_default(unique),
    uuid(FF9A1BB6-1E96-11d3-BFCC-00C04FA3490A)
]
interface ILinkQueue : IUnknown
{
    HRESULT GetInfo(
        [in, out] PQQUEUE_INFO pqi
    );

    HRESULT GetMessageEnum(
        [in] PMESSAGE_ENUM_FILTER pFilter,
        [out] IAQEnumMessages **ppenum
    );
}

[
    helpstring("Advanced Queue Administration Object"),
    object,
    pointer_default(unique),
    uuid(ba7af305-7373-11d2-94e4-00c04fa379f1)
]
interface IAQMessage : IUnknown
{
    HRESULT GetInfo(
        [in, out] PMESSAGE_INFO pmi
    );

    HRESULT GetContentStream(
        [out] IStream **ppIStream,
        [out] LPWSTR *pwszContentType
    );
}

[
    helpstring("Queue/Link Id Object"),
    object,
    pointer_default(unique),
    uuid(EA4DFDF2-9E87-4c57-B845-123872C5649F)
]
interface IUniqueId : IUnknown
{
    HRESULT GetUniqueId(

```

```
    [out] QUEUELINK_ID **ppqlid  
    );  
}
```

7 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft Windows® 2000 operating system
- Windows® XP operating system
- Windows Server® 2003 operating system
- Windows Vista® operating system
- Windows Server® 2008 operating system
- Windows Server® 2008 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

8 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

9 Index

A

Abstract data model

- IAQAdmin ([section 3.1.1](#) 20, [section 3.2.1](#) 21)
- IAQEnumMessages ([section 3.1.1](#) 20, [section 3.8.1](#) 37)
- IAQMessage ([section 3.1.1](#) 20, [section 3.9.1](#) 40)
- IAQMessageAction ([section 3.1.1](#) 20, [section 3.10.1](#) 43)
- IEnumLinkQueues ([section 3.1.1](#) 20, [section 3.6.1](#) 32)
- IEnumVSAQLinks ([section 3.1.1](#) 20, [section 3.4.1](#) 26)
- ILinkQueue ([section 3.1.1](#) 20, [section 3.7.1](#) 35)
- IUniqueId ([section 3.1.1](#) 20, [section 3.11.1](#) 45)
- IVSAQAdmin ([section 3.1.1](#) 20, [section 3.3.1](#) 23)
- IVSAQLink ([section 3.1.1](#) 20, [section 3.5.1](#) 29)

[Applicability](#) 9

[ApplyActionToMessages method](#) 43

[AQ_MESSAGE_FLAGS enumeration](#) 16

C

[Capability negotiation](#) 9

[Change tracking](#) 63

Clone method ([section 3.4.4.4](#) 28, [section 3.6.4.4](#) 34, [section 3.8.4.4](#) 39)

[Common data types](#) 10

D

Data model - abstract

- IAQAdmin ([section 3.1.1](#) 20, [section 3.2.1](#) 21)
- IAQEnumMessages ([section 3.1.1](#) 20, [section 3.8.1](#) 37)
- IAQMessage ([section 3.1.1](#) 20, [section 3.9.1](#) 40)
- IAQMessageAction ([section 3.1.1](#) 20, [section 3.10.1](#) 43)
- IEnumLinkQueues ([section 3.1.1](#) 20, [section 3.6.1](#) 32)
- IEnumVSAQLinks ([section 3.1.1](#) 20, [section 3.4.1](#) 26)
- ILinkQueue ([section 3.1.1](#) 20, [section 3.7.1](#) 35)
- IUniqueId ([section 3.1.1](#) 20, [section 3.11.1](#) 45)
- IVSAQAdmin ([section 3.1.1](#) 20, [section 3.3.1](#) 23)
- IVSAQLink ([section 3.1.1](#) 20, [section 3.5.1](#) 29)

[Data types](#) 10

E

Events

local

- IAQAdmin ([section 3.1.6](#) 21, [section 3.2.6](#) 22)
- IAQEnumMessages ([section 3.1.6](#) 21, [section 3.8.6](#) 40)
- IAQMessage ([section 3.1.6](#) 21, [section 3.9.6](#) 42)
- IAQMessageAction ([section 3.1.6](#) 21, [section 3.10.6](#) 45)

IEnumLinkQueues ([section 3.1.6](#) 21, [section 3.6.6](#) 34)

IEnumVSAQLinks ([section 3.1.6](#) 21, [section 3.4.6](#) 28)

ILinkQueue ([section 3.1.6](#) 21, [section 3.7.6](#) 37)

IUniqueId ([section 3.1.6](#) 21, [section 3.11.6](#) 46)

IVSAQAdmin ([section 3.1.6](#) 21, [section 3.3.6](#) 25)

IVSAQLink ([section 3.1.6](#) 21, [section 3.5.6](#) 31)

timer

IAQAdmin ([section 3.1.5](#) 21, [section 3.2.5](#) 22)

IAQEnumMessages ([section 3.1.5](#) 21, [section 3.8.5](#) 40)

IAQMessage ([section 3.1.5](#) 21, [section 3.9.5](#) 42)

IAQMessageAction ([section 3.1.5](#) 21, [section 3.10.5](#) 45)

IEnumLinkQueues ([section 3.1.5](#) 21, [section 3.6.5](#) 34)

IEnumVSAQLinks ([section 3.1.5](#) 21, [section 3.4.5](#) 28)

ILinkQueue ([section 3.1.5](#) 21, [section 3.7.5](#) 36)

IUniqueId ([section 3.1.5](#) 21, [section 3.11.5](#) 46)

IVSAQAdmin ([section 3.1.5](#) 21, [section 3.3.5](#) 25)

IVSAQLink ([section 3.1.5](#) 21, [section 3.5.5](#) 31)

[Examples - overview](#) 47

F

[Fields - vendor-extensible](#) 9

[Full IDL](#) 52

G

[GetContentStream method](#) 41

[GetGlobalLinkState method](#) 25

[GetInfo method](#) ([section 3.5.4.1](#) 29, [section 3.7.4.1](#) 35, [section 3.9.4.1](#) 40)

[GetLinkEnum method](#) 23

[GetMessageEnum method](#) 36

[GetQueueEnum method](#) 31

[GetUniqueId method](#) 45

[GetVirtualServerAdminITF method](#) 21

[Glossary](#) 7

I

IAQAdmin

abstract data model ([section 3.1.1](#) 20, [section 3.2.1](#) 21)

[GetVirtualServerAdminITF method](#) 21

initialization ([section 3.1.3](#) 20, [section 3.2.3](#) 21)

local events ([section 3.1.6](#) 21, [section 3.2.6](#) 22)

[message processing](#) 20

[sequencing rules](#) 20

timer events ([section 3.1.5](#) 21, [section 3.2.5](#) 22)

timers ([section 3.1.2](#) 20, [section 3.2.2](#) 21)

IAQEnumMessages

abstract data model ([section 3.1.1](#) 20, [section 3.8.1](#) 37)
[Clone method](#) 39
 initialization ([section 3.1.3](#) 20, [section 3.8.3](#) 37)
 local events ([section 3.1.6](#) 21, [section 3.8.6](#) 40)
[message processing](#) 20
[Next method](#) 37
[Reset method](#) 39
[sequencing rules](#) 20
[Skip method](#) 38
 timer events ([section 3.1.5](#) 21, [section 3.8.5](#) 40)
 timers ([section 3.1.2](#) 20, [section 3.8.2](#) 37)

IAQMessage
 abstract data model ([section 3.1.1](#) 20, [section 3.9.1](#) 40)
[GetContentStream method](#) 41
[GetInfo method](#) 40
 initialization ([section 3.1.3](#) 20, [section 3.9.3](#) 40)
 local events ([section 3.1.6](#) 21, [section 3.9.6](#) 42)
[message processing](#) 20
[sequencing rules](#) 20
 timer events ([section 3.1.5](#) 21, [section 3.9.5](#) 42)
 timers ([section 3.1.2](#) 20, [section 3.9.2](#) 40)

IAQMessageAction
 abstract data model ([section 3.1.1](#) 20, [section 3.10.1](#) 43)
[ApplyActionToMessages method](#) 43
 initialization ([section 3.1.3](#) 20, [section 3.10.3](#) 43)
 local events ([section 3.1.6](#) 21, [section 3.10.6](#) 45)
[message processing](#) 20
[QuerySupportedActions method](#) 44
[sequencing rules](#) 20
 timer events ([section 3.1.5](#) 21, [section 3.10.5](#) 45)
 timers ([section 3.1.2](#) 20, [section 3.10.2](#) 43)

[IDL](#) 52

IEnumLinkQueues
 abstract data model ([section 3.1.1](#) 20, [section 3.6.1](#) 32)
[Clone method](#) 34
 initialization ([section 3.1.3](#) 20, [section 3.6.3](#) 32)
 local events ([section 3.1.6](#) 21, [section 3.6.6](#) 34)
[message processing](#) 20
[Next method](#) 32
[Reset method](#) 33
[sequencing rules](#) 20
[Skip method](#) 33
 timer events ([section 3.1.5](#) 21, [section 3.6.5](#) 34)
 timers ([section 3.1.2](#) 20, [section 3.6.2](#) 32)

IEnumVSAQLinks
 abstract data model ([section 3.1.1](#) 20, [section 3.4.1](#) 26)
[Clone method](#) 28
 initialization ([section 3.1.3](#) 20, [section 3.4.3](#) 26)
 local events ([section 3.1.6](#) 21, [section 3.4.6](#) 28)
[message processing](#) 20
[Next method](#) 26
[Reset method](#) 27
[sequencing rules](#) 20
[Skip method](#) 27
 timer events ([section 3.1.5](#) 21, [section 3.4.5](#) 28)

timers ([section 3.1.2](#) 20, [section 3.4.2](#) 26)

ILinkQueue
 abstract data model ([section 3.1.1](#) 20, [section 3.7.1](#) 35)
[GetInfo method](#) 35
[GetMessageEnum method](#) 36
 initialization ([section 3.1.3](#) 20, [section 3.7.3](#) 35)
 local events ([section 3.1.6](#) 21, [section 3.7.6](#) 37)
[message processing](#) 20
[sequencing rules](#) 20
 timer events ([section 3.1.5](#) 21, [section 3.7.5](#) 36)
 timers ([section 3.1.2](#) 20, [section 3.7.2](#) 35)

[Implementer - security considerations](#) 51
[Index of security parameters](#) 51
[Informative references](#) 8

Initialization
 IAQAdmin ([section 3.1.3](#) 20, [section 3.2.3](#) 21)
 IAQEnumMessages ([section 3.1.3](#) 20, [section 3.8.3](#) 37)
 IAQMessage ([section 3.1.3](#) 20, [section 3.9.3](#) 40)
 IAQMessageAction ([section 3.1.3](#) 20, [section 3.10.3](#) 43)
 IEnumLinkQueues ([section 3.1.3](#) 20, [section 3.6.3](#) 32)
 IEnumVSAQLinks ([section 3.1.3](#) 20, [section 3.4.3](#) 26)
 ILinkQueue ([section 3.1.3](#) 20, [section 3.7.3](#) 35)
 IUniqueId ([section 3.1.3](#) 20, [section 3.11.3](#) 45)
 IVSAQAdmin ([section 3.1.3](#) 20, [section 3.3.3](#) 23)
 IVSAQLink ([section 3.1.3](#) 20, [section 3.5.3](#) 29)

[Introduction](#) 7

IUniqueId
 abstract data model ([section 3.1.1](#) 20, [section 3.11.1](#) 45)
[GetUniqueId method](#) 45
 initialization ([section 3.1.3](#) 20, [section 3.11.3](#) 45)
 local events ([section 3.1.6](#) 21, [section 3.11.6](#) 46)
[message processing](#) 20
[sequencing rules](#) 20
 timer events ([section 3.1.5](#) 21, [section 3.11.5](#) 46)
 timers ([section 3.1.2](#) 20, [section 3.11.2](#) 45)

IVSAQAdmin
 abstract data model ([section 3.1.1](#) 20, [section 3.3.1](#) 23)
[GetGlobalLinkState method](#) 25
[GetLinkEnum method](#) 23
 initialization ([section 3.1.3](#) 20, [section 3.3.3](#) 23)
 local events ([section 3.1.6](#) 21, [section 3.3.6](#) 25)
[message processing](#) 20
[sequencing rules](#) 20
[StartAllLinks method](#) 24
[StopAllLinks method](#) 24
 timer events ([section 3.1.5](#) 21, [section 3.3.5](#) 25)
 timers ([section 3.1.2](#) 20, [section 3.3.2](#) 23)

IVSAQLink
 abstract data model ([section 3.1.1](#) 20, [section 3.5.1](#) 29)
[GetInfo method](#) 29
[GetQueueEnum method](#) 31
 initialization ([section 3.1.3](#) 20, [section 3.5.3](#) 29)

local events ([section 3.1.6](#) 21, [section 3.5.6](#) 31)
[message processing](#) 20
[sequencing rules](#) 20
[SetLinkState](#) method 30
timer events ([section 3.1.5](#) 21, [section 3.5.5](#) 31)
timers ([section 3.1.2](#) 20, [section 3.5.2](#) 29)

L

[LINK_ACTION enumeration](#) 15
[LINK_INFO structure](#) 15
[LINK_INFO_FLAGS enumeration](#) 14

Local events

IAQAdmin ([section 3.1.6](#) 21, [section 3.2.6](#) 22)
IAQEnumMessages ([section 3.1.6](#) 21, [section 3.8.6](#) 40)
IAQMessage ([section 3.1.6](#) 21, [section 3.9.6](#) 42)
IAQMessageAction ([section 3.1.6](#) 21, [section 3.10.6](#) 45)
IEnumLinkQueues ([section 3.1.6](#) 21, [section 3.6.6](#) 34)
IEnumVSAQLinks ([section 3.1.6](#) 21, [section 3.4.6](#) 28)
ILinkQueue ([section 3.1.6](#) 21, [section 3.7.6](#) 37)
IUniqueId ([section 3.1.6](#) 21, [section 3.11.6](#) 46)
IVSAQAdmin ([section 3.1.6](#) 21, [section 3.3.6](#) 25)
IVSAQLink ([section 3.1.6](#) 21, [section 3.5.6](#) 31)

M

Message processing

[IAQAdmin](#) 20
[IAQEnumMessages](#) 20
[IAQMessage](#) 20
[IAQMessageAction](#) 20
[IEnumLinkQueues](#) 20
[IEnumVSAQLinks](#) 20
[ILinkQueue](#) 20
[IUniqueId](#) 20
[IVSAQAdmin](#) 20
[IVSAQLink](#) 20

[MESSAGE_ACTION enumeration](#) 12
[MESSAGE_ENUM_FILTER structure](#) 13
[MESSAGE_ENUM_FILTER_TYPE enumeration](#) 12
[MESSAGE_FILTER structure](#) 10
[MESSAGE_FILTER_FLAGS enumeration](#) 11
[MESSAGE_INFO structure](#) 17

Messages

[data types](#) 10
[transport](#) 10

Methods

[ApplyActionToMessages](#) 43
Clone ([section 3.4.4.4](#) 28, [section 3.6.4.4](#) 34, [section 3.8.4.4](#) 39)
[GetContentStream](#) 41
[GetGlobalLinkState](#) 25
GetInfo ([section 3.5.4.1](#) 29, [section 3.7.4.1](#) 35, [section 3.9.4.1](#) 40)
[GetLinkEnum](#) 23
[GetMessageEnum](#) 36
[GetQueueEnum](#) 31
[GetUniqueId](#) 45

[GetVirtualServerAdminITF](#) 21
Next ([section 3.4.4.1](#) 26, [section 3.6.4.1](#) 32, [section 3.8.4.1](#) 37)
[QuerySupportedActions](#) 44
Reset ([section 3.4.4.3](#) 27, [section 3.6.4.3](#) 33, [section 3.8.4.3](#) 39)
[SetLinkState](#) 30
Skip ([section 3.4.4.2](#) 27, [section 3.6.4.2](#) 33, [section 3.8.4.2](#) 38)
[StartAllLinks](#) 24
[StopAllLinks](#) 24

N

Next method ([section 3.4.4.1](#) 26, [section 3.6.4.1](#) 32, [section 3.8.4.1](#) 37)
[Normative references](#) 8

O

[Overview \(synopsis\)](#) 8

P

[Parameters - security index](#) 51
[PLINK_INFO](#) 15
[PMESSAGE_ENUM_FILTER](#) 13
[PMESSAGE_FILTER](#) 10
[PMESSAGE_INFO](#) 17
[PQUEUE_INFO](#) 16
[Preconditions](#) 9
[Prerequisites](#) 9
[Product behavior](#) 62

Q

[QuerySupportedActions](#) method 44
[QUEUE_ADMIN_VERSIONS enumeration](#) 10
[QUEUE_INFO structure](#) 16
[QUEUELINK_ID structure](#) 18
[QUEUELINK_TYPE enumeration](#) 18

R

References

[informative](#) 8
[normative](#) 8
[Relationship to other protocols](#) 9
Reset method ([section 3.4.4.3](#) 27, [section 3.6.4.3](#) 33, [section 3.8.4.3](#) 39)

S

Security

[implementer considerations](#) 51
[parameter index](#) 51

Sequencing rules

[IAQAdmin](#) 20
[IAQEnumMessages](#) 20
[IAQMessage](#) 20
[IAQMessageAction](#) 20
[IEnumLinkQueues](#) 20

[IEnumVSAQLinks](#) 20
[ILinkQueue](#) 20
[IUniqueId](#) 20
[IVSAQAdmin](#) 20
[IVSAQLink](#) 20
[SetLinkState method](#) 30
Skip method ([section 3.4.4.2](#) 27, [section 3.6.4.2](#) 33, [section 3.8.4.2](#) 38)
[Standards assignments](#) 9
[StartAllLinks method](#) 24
[StopAllLinks method](#) 24

T

Timer events
IAQAdmin ([section 3.1.5](#) 21, [section 3.2.5](#) 22)
IAQEnumMessages ([section 3.1.5](#) 21, [section 3.8.5](#) 40)
IAQMessage ([section 3.1.5](#) 21, [section 3.9.5](#) 42)
IAQMessageAction ([section 3.1.5](#) 21, [section 3.10.5](#) 45)
IEnumLinkQueues ([section 3.1.5](#) 21, [section 3.6.5](#) 34)
IEnumVSAQLinks ([section 3.1.5](#) 21, [section 3.4.5](#) 28)
ILinkQueue ([section 3.1.5](#) 21, [section 3.7.5](#) 36)
IUniqueId ([section 3.1.5](#) 21, [section 3.11.5](#) 46)
IVSAQAdmin ([section 3.1.5](#) 21, [section 3.3.5](#) 25)
IVSAQLink ([section 3.1.5](#) 21, [section 3.5.5](#) 31)
Timers
IAQAdmin ([section 3.1.2](#) 20, [section 3.2.2](#) 21)
IAQEnumMessages ([section 3.1.2](#) 20, [section 3.8.2](#) 37)
IAQMessage ([section 3.1.2](#) 20, [section 3.9.2](#) 40)
IAQMessageAction ([section 3.1.2](#) 20, [section 3.10.2](#) 43)
IEnumLinkQueues ([section 3.1.2](#) 20, [section 3.6.2](#) 32)
IEnumVSAQLinks ([section 3.1.2](#) 20, [section 3.4.2](#) 26)
ILinkQueue ([section 3.1.2](#) 20, [section 3.7.2](#) 35)
IUniqueId ([section 3.1.2](#) 20, [section 3.11.2](#) 45)
IVSAQAdmin ([section 3.1.2](#) 20, [section 3.3.2](#) 23)
IVSAQLink ([section 3.1.2](#) 20, [section 3.5.2](#) 29)
[Tracking changes](#) 63
[Transport](#) 10

V

[Vendor-extensible fields](#) 9
[Versioning](#) 9