[MS-IRP-Diff]:

Internet Information Services (IIS) Inetinfo Remote Protocol

Revision Summary

Date	Revision History	Revision Class	Comments
7/20/2007	0.1	Major	MCPP Milestone 5 Initial Availability
9/28/2007	0.2	Minor	Clarified the meaning of the technical content.
10/23/2007	0.2.1	Editorial	Changed language and formatting in the technical content.
11/30/2007	0.2.2	Editorial	Changed language and formatting in the technical content.
1/25/2008	0.2.3	Editorial	Changed language and formatting in the technical content.
3/14/2008	1.0	Major	Updated and revised the technical content.
5/16/2008	1.0.1	Editorial	Changed language and formatting in the technical content.
6/20/2008	1.1	Minor	Clarified the meaning of the technical content.
7/25/2008	1.2	Minor	Clarified the meaning of the technical content.
8/29/2008	2.0	Major	Updated and revised the technical content.
10/24/2008	2.0.1	Editorial	Changed language and formatting in the technical content.
12/5/2008	2.0.2	Editorial	Changed language and formatting in the technical content.
1/16/2009	3.0	Major	Updated and revised the technical content.
2/27/2009	3.0.1	Editorial	Changed language and formatting in the technical content.
4/10/2009	3.0.2	Editorial	Changed language and formatting in the technical content.
5/22/2009	3.0.3	Editorial	Changed language and formatting in the technical content.
7/2/2009	3.0.4	Editorial	Changed language and formatting in the technical content.
8/14/2009	3.1	Minor	Clarified the meaning of the technical content.
9/25/2009	3.2	Minor	Clarified the meaning of the technical content.
11/6/2009	3.2.1	Editorial	Changed language and formatting in the technical content.
12/18/2009	3.2.2	Editorial	Changed language and formatting in the technical content.
1/29/2010	4.0	Major	Updated and revised the technical content.
3/12/2010	4.0.1	Editorial	Changed language and formatting in the technical content.
4/23/2010	4.0.2	Editorial	Changed language and formatting in the technical content.
6/4/2010	4.0.3	Editorial	Changed language and formatting in the technical content.
7/16/2010	4.0.3	None	No changes to the meaning, language, or formatting of the technical content.
8/27/2010	4.0.3	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2010	4.0.3	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
11/19/2010	4.0.3	None	No changes to the meaning, language, or formatting of the technical content.
1/7/2011	4.0.3	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2011	4.0.3	None	No changes to the meaning, language, or formatting of the technical content.
3/25/2011	4.0.3	None	No changes to the meaning, language, or formatting of the technical content.
5/6/2011	4.0.3	None	No changes to the meaning, language, or formatting of the technical content.
6/17/2011	4.1	Minor	Clarified the meaning of the technical content.
9/23/2011	4.1	None	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	5.0	Major	Updated and revised the technical content.
3/30/2012	5.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	5.0	None	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	5.0	None	No changes to the meaning, language, or formatting of the technical content.
1/31/2013	5.0	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	6.0	Major	Updated and revised the technical content.
11/14/2013	6.0	None	No changes to the meaning, language, or formatting of the technical content.
2/13/2014	6.0	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	6.0	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	7.0	Major	Significantly changed the technical content.
10/16/2015	7.0	None	No changes to the meaning, language, or formatting of the technical content.
7/14/2016	7.0	None	No changes to the meaning, language, or formatting of the technical content.
6/1/2017	8.0	Major	Significantly changed the technical content.
9/15/2017	9.0	Major	Significantly changed the technical content.
12/1/2017	9.0	None	No changes to the meaning, language, or formatting of the technical content.
9/12/2018	10.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The Internet Information Services (IIS) Inetinfo Remote Protocol is a remote procedure call (RPC)–based client/server protocol that is used for managing Internet protocol servers such as those hosted by Microsoft Internet Information Services (IIS). Managed servers can include servers for HTTP, FTP, SMTP, or other Internet protocols. For more information on IIS, see [MSDN-IIS].

The universally unique identifier (UUID) for the IIS Inetinfo Remote Protocol interface is {82ad4280-036b-11cf-972c-00aa006887b0}.

The version for this interface is 2.0.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

authentication level: A numeric value indicating the level of authentication or message protection that remote procedure call (RPC) will apply to a specific message exchange. For more information, see [C706] section 13.1.2.1 and [MS-RPCE].

Authentication Service (AS): A service that issues ticket granting tickets (TGTs), which are used for authenticating principals within the realm or domain served by the Authentication Service.

Binary Gateway Interface (BGI): An extension API for HTTP servers that is analogous to the Common Gateway Interface (CGI) but relies on direct method calls and parameter passing. In the IIS HTTP server, BGI is equivalent to the Internet Server API (ISAPI).

binary large object (BLOB): A discrete packet of data that is stored in a database and is treated as a sequence of uninterpreted bytes.

endpoint: (1) A client that is on a network and is requesting access to a network access server (NAS).

(2) A network-specific address of a remote procedure call (RPC) server process for remote procedure calls. The actual name and type of the endpoint depends on the RPC protocol sequence that is being used. For example, for RPC over TCP (RPC Protocol Sequence ncacn_ip_tcp), an endpoint might be TCP port 1025. For RPC over Server Message Block (RPC Protocol Sequence ncacn_np), an endpoint might be the name of a named pipe. For more information, see [C706].

Interface Definition Language (IDL): The International Standards Organization (ISO) standard language for specifying the interface for remote procedure calls. For more information, see [C706] section 4.

Internet Information Services (IIS): The services provided in Windows implementation that support web server functionality. IIS consists of a collection of standard Internet protocol servers such as HTTP and FTP in addition to common infrastructures that are used by other Microsoft Internet protocol servers such as SMTP, NNTP, and so on. IIS has been part of the Windows operating system in some versions and a separate install package in others. IIS version 5.0 shipped as part of Windows 2000 operating system, IIS version 5.1 as part of Windows XP operating system, IIS version 6.0 as part of Windows Server 2003 operating system, and IIS version 7.0 as part of Windows Vista operating system and Windows Server 2008 operating system.

Internet protocol server: A software program that implements the server host of a standard Internet protocol such as HTTP or FTP.

Internet protocol server instance (server instance): A configuration collection for an Internet protocol server that will establish its own network protocol endpoint. A single Internet protocol server may configure multiple server instances that would each appear to clients as an independent host (also referred to as a site).

network byte order: The order in which the bytes of a multiple-byte number are transmitted on a network, most significant byte first (in big-endian storage). This may or may not match the order in which numbers are normally stored in memory for a particular processor.

opnum: An operation number or numeric identifier that is used to identify a specific remote procedure call (RPC) method or a method in an interface. For more information, see [C706] section 12.5.2.12 or [MS-RPCE].

remote procedure call (RPC): A communication protocol used primarily between client and server. The term has three definitions that are often used interchangeably: a runtime environment providing for communication facilities between computers (the RPC runtime); a set of request-and-response message exchanges between computers (the RPC exchange); and the single message from an RPC exchange (the RPC message). For more information, see [C706].

RPC client: A computer on the network that sends messages using remote procedure call (RPC) as its transport, waits for responses, and is the initiator in an RPC exchange.

RPC protocol sequence: A character string that represents a valid combination of a remote procedure call (RPC) protocol, a network layer protocol, and a transport layer protocol, as described in [C706] and [MS-RPCE].

Server Message Block (SMB): A protocol that is used to request file and print services from server systems over a network. The SMB protocol extends the CIFS protocol with additional security, file, and disk management support. For more information, see [CIFS] and [MS-SMB].

stub: Used as specified in [C706] section 2.1.2.2. A stub that is used on the client is called a "client stub", and a stub that is used on the server is called a "server stub".

universally unique identifier (UUID): A 128-bit value. UUIDs can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects in cross-process communication such as client and server interfaces, manager entry-point vectors, and RPC objects. UUIDs are highly likely to be unique. UUIDs are also known as globally unique identifiers (GUIDs) and these terms are used interchangeably in the Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the UUID. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the UUID.

virtual root: A configured mapping within an Internet protocol server instance between an instance URI and a file system directory. For example, a virtual root could map the URI "/somepath" to the file system directory "d:\webcontent". For more information about the syntax of a URI, see [RFC3986].

well-known endpoint: A preassigned, network-specific, stable address for a particular client/server instance. For more information, see [C706].

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents

in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

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

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

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

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

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

1.2.2 (Updated Section) Informative References

[MSDN-IIS] Microsoft Corporation, "Internet Information Services (IIS)", http://msdn.microsoft.com/en-us/library/aa286507.aspx

[MSDN-MIDL] Microsoft Corporation, "Microsoft Interface Definition Language (MIDL)", http://msdn.microsoft.com/en-us/library/ms950375.aspx

[MSFT-CAL] Microsoft Corporation, "Client Access Licenses ~~(CALs)", http~~and Management Licenses", https://www.microsoft.com/~~resources/sam/lic_cal.mspx~~en-us/licensing/product-licensing/client-access-license.aspx

[RFC2068] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997, http://www.ietf.org/rfc/rfc2068.txt

[RFC2518] Goland, Y., Whitehead, E., Faizi, A., et al., "HTTP Extensions for Distributed Authoring - WebDAV", RFC 2518, February 1999, http://www.ietf.org/rfc/rfc2518.txt

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.rfc-editor.org/rfc/rfc2818.txt

[RFC3875] Robinson, D., and Coar, K., "The Common Gateway Interface (CGI) Version 1.1", October 2004, http://www.ietf.org/rfc/rfc3875

1.3 Overview

The Internet Information Services (IIS) Inetinfo Remote Protocol provides functions that allow remote administration and statistics gathering from an Internet protocol server such as a server implementing the HTTP or FTP protocol. The protocol provides methods for gathering statistical data on users, sites, requests, and performance. For more information about HTTP and securing HTTP connections, see [RFC2068] and [RFC2818].

The server does not maintain client state information. Although some client call sequences might be logically related, the protocol operation is stateless.

1.4 Relationship to Other Protocols

The Internet Information Services (IIS) Inetinfo Remote Protocol uses RPC as its protocol transport, as specified in [MS-RPCE].

1.5 Prerequisites/Preconditions

This protocol requires that the client and server be able to communicate by means of an RPC connection, as specified in section 2.1.

1.6 Applicability Statement

The Internet Information Services (IIS) Inetinfo Remote Protocol is appropriate for managing an Internet protocol server or a collection of such servers on a remote computer.

1.7 Versioning and Capability Negotiation

The Internet Information Services (IIS) Inetinfo Remote Protocol has been modified between versions of IIS in ways that make interoperability between different server implementations difficult. Modifications to the interface between IIS versions will be noted in section 2.2 or section 3.1.<1>

1.8 Vendor-Extensible Fields

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

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The Internet Information Services (IIS) Inetinfo Remote Protocol MUST use RPC as the transport protocol.

2.1.1 Server

The server interface MUST be identified by UUID "82ad4280-036b-11cf-972c-00aa006887b0", version 2.0.

The server MUST specify RPC over Server Message Block (SMB) as the RPC protocol sequence to the RPC implementation (as specified in [MS-RPCE] section 2.1.1.2), using the RPC well-known endpoint \PIPE\inetinfo.

The server MUST also specify RPC over TCP/IP as an RPC protocol sequence to the RPC implementation, as specified in [MS-RPCE] section 2.1.1.1.

The server SHOULD specify "NTLM" (0xA) as the RPC authentication service, as specified in [MS-RPCE] section 3.<2>

2.1.2 Client

The client SHOULD use RPC over SMB (ncacn_np) or RPC over TCP/IP (ncacn_ip_tcp) as the RPC protocol sequence to communicate with the server. Using other protocol sequences MAY work depending on the configuration and implementation of the server.

The client MAY use an authentication level of privacy to connect to the server and if the server does not support this authentication level, it MAY fall back to connection. Authentication levels are specified in [MS-RPCE].

2.2 Common Data Types

In addition to the RPC data types that are specified in [MS-RPCE], the sections that follow use the definitions of DWORD, WCHAR, LPWSTR, LCID, LARGE_INTEGER, and BYTE, as specified in [MS-DTYP].

For more information on the additional data types in the following sections, see [MSDN-MIDL].

2.2.1 INET_INFO_IMPERSONATE_HANDLE

The INET_INFO_IMPERSONATE_HANDLE type defines a pointer to an array of WCHAR elements. The client uses this pointer as a custom binding handle, which it converts to an explicit server binding handle for a target server. On the client, the value MUST be sufficient to generate an appropriate server binding handle. The value of this parameter MUST NOT be used on the server.

This type is declared as follows:

```
typedef [handle] [string] LPWSTR INET_INFO_IMPERSONATE_HANDLE;
```

2.2.2 Internet Protocol Server Identifiers

The service identifiers listed below are used by several methods of the Internet Information Services (IIS) Inetinfo Remote Protocol to indicate the type of Internet protocol server targeted by the method. Server implementations MAY implement the Internet Information Services (IIS) Inetinfo Remote Protocol for some or all of the server types specified below.

If the specified Internet protocol server is not being managed by an Internet Information Services (IIS) Inetinfo Remote Protocol implementation or if the server is unavailable or inactive, method calls that target that server SHOULD return ERROR_SERVICE_NOT_ACTIVE, as specified in section 2.2.7.

Constant/value	Description
INET_FTP_SVC_ID 0x00000001	Identifies the File Transfer Protocol (FTP) service.
INET_GOPHER_SVC_ID 0x00000002	Identifies the Gopher service.
INET_HTTP_SVC_ID 0x00000004	Identifies the Hypertext Transfer Protocol (HTTP) service.
INET_DNS_SVC_ID 0x00000008	Identifies the Domain Name System (DNS).
INET_HTTP_PROXY 0x00000010	Identifies the HTTP proxy service.
INET_NNTP_SVC_ID 0x00000040	Identifies the Network News Transfer Protocol (NNTP) service.
INET_SMTP_SVC_ID 0x00000080	Identifies the Simple Mail Transfer Protocol (SMTP) service.
INET_GATEWAY_SVC_ID 0x00000100	Identifies the Internet gateway service.
INET_POP3_SVC_ID 0x00000200	Identifies the Post Office Protocol 3 (POP3) service.
INET_CHAT_SVC_ID 0x00000400	Identifies the Internet Relay Chat (IRC) service.
INET_LDAP_SVC_ID 0x00000800	Identifies the Lightweight Directory Access Protocol (LDAP) service.
INET_IMAP_SVC_ID 0x00001000	Identifies the Internet Message Access Protocol (IMAP) service.

2.2.3 INET_INFO_CONFIG_INFO

The **INET_INFO_CONFIG_INFO** structure stores configuration values for an Internet protocol server.<3>

This type is declared as follows:

```
typedef struct _INET_INFO_CONFIG_INFO {
  DWORD FieldControl;
```

```
  DWORD dwConnectionTimeout;
  DWORD dwMaxConnections;
  [string] LPWSTR lpszAdminName;
  [string] LPWSTR lpszAdminEmail;
  [string] LPWSTR lpszServerComment;
  LPINET_LOG_CONFIGURATION lpLogConfig;
  WORD LangId;
  LCID LocalId;
  BYTE ProductId[64];
  BOOL fLogAnonymous;
  BOOL fLogNonAnonymous;
  [string] LPWSTR lpszAnonUserName;
  WCHAR szAnonPassword[257];
  DWORD dwAuthentication;
  short sPort;
  LPINET_INFO_IP_SEC_LIST DenyIPList;
  LPINET_INFO_IP_SEC_LIST GrantIPList;
  LPINET_INFO_VIRTUAL_ROOT_LIST VirtualRoots;
} INET_INFO_CONFIG_INFO,
 *LPINET_INFO_CONFIG_INFO;
```

FieldControl: A 32-bit unsigned integer that specifies a bit field. This field specifies the values of the **INET_INFO_CONFIG_INFO** structure that contain valid data. An implementation MUST set the flag corresponding to the structure field when returning or updating configuration data. This field MUST be set to a valid combination of the following values.

Name	Value
FC_INET_INFO_CONNECTION_TIMEOUT	0x00000001
FC_INET_INFO_MAX_CONNECTIONS	0x00000002
FC_INET_INFO_LOG_CONFIG	0x00000004
FC_INET_INFO_ADMIN_NAME	0x00000008
FC_INET_INFO_SERVER_COMMENT	0x00000010
FC_INET_INFO_ADMIN_EMAIL	0x00000020
FC_INET_INFO_HOST_NAME	0x00000040
FC_INET_INFO_SERVER_SIZE	0x00000080
FC_INET_INFO_DEF_LOGON_DOMAIN	0x00008000
FC_INET_INFO_AUTHENTICATION	0x00010000
FC_INET_INFO_ALLOW_ANONYMOUS	0x00020000
FC_INET_INFO_LOG_ANONYMOUS	0x00040000
FC_INET_INFO_LOG_NONANONYMOUS	0x00080000
FC_INET_INFO_ANON_USER_NAME	0x00100000
FC_INET_INFO_ANON_PASSWORD	0x00200000
FC_INET_INFO_PORT_NUMBER	0x00400000
FC_INET_INFO_SITE_SECURITY	0x00800000
FC_INET_INFO_VIRTUAL_ROOTS	0x01000000
FC_INET_INFO_SECURE_PORT_NUMBER	0x02000000

Name	Value
FC_INET_INFO_SERVER_NAME	0x04000000
FC_INET_INFO_AUTO_START	0x08000000
FC_INET_INFO_ADDRESS_TYPE	0x10000000
FC_INET_INFO_IP_ADDRESS	0x20000000

dwConnectionTimeout: The time limit to maintain an inactive connection specified as the number of seconds from the last request.

dwMaxConnections: The maximum number of allowed connections to the Internet protocol server.

lpszAdminName: A pointer to a null-terminated Unicode string that contains the name of the Internet protocol server administrator.

lpszAdminEmail: A pointer to a null-terminated Unicode string that contains the email address of the Internet protocol server administrator.

lpszServerComment: A pointer to a null-terminated Unicode string that contains a comment that describes the Internet protocol server instance.

lpLogConfig: A pointer to an INET_LOG_CONFIGURATION structure that specifies the configuration of the Internet protocol server log.

LangId: A WORD containing the language identifier, a standard international numerical identifier for the language in the country or region where the server is in use, as specified in [MS-LCID] section 2.1.

LocalId: A locale identifier that is a DWORD value that consists of a language identifier, such as one specified for the **LangID** member, combined with a sort identifier indicating location. For example, the **LangID** member might indicate French, where the **LocalID** indicates Canada. The **LocalID** member is given as specified in [MS-LCID] section 2.1.

ProductId: An array of 64 bytes that MAY contain a string value, which identifies the server implementation.

fLogAnonymous: A value that is set to TRUE if data transfers from anonymous users are to be logged.

fLogNonAnonymous: A value that is set to TRUE if data transfers from non-anonymous users are to be logged.

lpszAnonUserName: A pointer to a null-terminated Unicode string that contains the name requested and accepted from an anonymous user.

szAnonPassword: An array of 257 Unicode characters containing a null-terminated Unicode string that contains a password requested and accepted from an anonymous user.<4>

dwAuthentication: A value that indicates what authentication methods can be used.

sPort: A 16-bit unsigned integer that specifies the network port on which the Internet protocol server instance is running.

DenyIPList: A pointer to an INET_INFO_IP_SEC_LIST structure that contains a list of IP addresses that will be denied connections to the Internet protocol server.

GrantIPList: A pointer to an **INET_INFO_IP_SEC_LIST** structure that contains a list of IP addresses that will be granted connections to the Internet protocol server.

VirtualRoots: A pointer to an INET_INFO_VIRTUAL_ROOT_LIST structure that contains a list of virtual root directories for the Internet protocol server instance.

2.2.4 INET_LOG_CONFIGURATION

The **INET_LOG_CONFIGURATION** structure contains configuration information for Internet protocol server logging.

This type is declared as follows:

```
typedef struct _INET_LOG_CONFIGURATION {
  DWORD inetLogType;
  DWORD ilPeriod;
  WCHAR rgchLogFileDirectory[260];
  DWORD cbSizeForTruncation;
  WCHAR rgchDataSource[260];
  WCHAR rgchTableName[30];
  WCHAR rgchUserName[257];
  WCHAR rgchPassword[257];
} INET_LOG_CONFIGURATION,
 *LPINET_LOG_CONFIGURATION;
```

inetLogType: A 32-bit integer that specifies the type of log to be written. This field MUST be set to one of the following values.

Value	Meaning
INET_LOG_DISABLED 0x00000000	Logging is disabled.
INET_LOG_TO_FILE 0x00000001	The log is written to a file.
INET_LOG_TO_SQL 0x00000002	The log is written to a Structured Query Language (SQL) database.
INET_LOG_INVALID 0xFFFFFFFF	The log is not valid.

ilPeriod: Specifies the periodicity of Internet protocol server logging. This field MUST be set to one of the following values.

Value	Meaning
INET_LOG_PERIOD_NONE 0x00000000	There is no log period.
INET_LOG_PERIOD_DAILY 0x00000001	The log period is daily.
INET_LOG_PERIOD_WEEKLY 0x00000002	The log period is weekly.
INET_LOG_PERIOD_MONTHLY 0x00000003	The log period is monthly.
INET_LOG_PERIOD_HOURLY 0x00000004	The log period is hourly.

Value	Meaning
INET_LOG_PERIOD_YEARLY 0x00000005	The log period is yearly.
INET_LOG_PERIOD_INVALID 0xFFFFFFFF	The log period is not valid.

rgchLogFileDirectory: A null-terminated string that specifies the destination of the Internet protocol server log.

cbSizeForTruncation: Specifies the maximum size in bytes for each log file.

rgchDataSource: A null-terminated string that specifies the Open Database Connectivity (ODBC) data source name to which the Internet protocol server log is to be written.

rgchTableName: A null-terminated string that specifies the name of the table on **rgchDataSource** to which the Internet protocol server log is to be written.

rgchUserName: A null-terminated string that specifies the name of the user for the ODBC connection.

rgchPassword: A null-terminated string that specifies the password associated with the **rgchUserName** user name.

2.2.5 INET_INFO_IP_SEC_LIST

The **INET_INFO_IP_SEC_LIST** structure contains a list of INET_INFO_IP_SEC_ENTRY entries.

This type is declared as follows:

```
typedef struct _INET_INFO_IP_SEC_LIST {
  DWORD cEntries;
  [size_is(cEntries)] INET_INFO_IP_SEC_ENTRY aIPSecEntry[];
} INET_INFO_IP_SEC_LIST,
 *LPINET_INFO_IP_SEC_LIST;
```

cEntries: The number of entries contained in the list.

aIPSecEntry: An array of **INET_INFO_IP_SEC_ENTRY** entries.

2.2.6 INET_INFO_IP_SEC_ENTRY

The **INET_INFO_IP_SEC_ENTRY** structure contains Internet Protocol Security (IPv4) entries.

This type is declared as follows:

```
typedef struct _INET_INFO_IP_SEC_ENTRY {
  DWORD dwMask;
  DWORD dwNetwork;
} INET_INFO_IP_SEC_ENTRY,
 *LPINET_INFO_IP_SEC_ENTRY;
```

dwMask: The subnet mask. Data is stored in network byte order.

dwNetwork: The IP address. Data is stored in network byte order.

2.2.7 INET_INFO_VIRTUAL_ROOT_LIST

The **INET_INFO_VIRTUAL_ROOT_LIST** structure contains a list of INET_INFO_VIRTUAL_ROOT_ENTRY virtual root entries.

This type is declared as follows:

```
typedef struct _INET_INFO_VIRTUAL_ROOT_LIST {
  DWORD cEntries;
  [size_is(cEntries)] INET_INFO_VIRTUAL_ROOT_ENTRY aVirtRootEntry[];
} INET_INFO_VIRTUAL_ROOT_LIST,
 *LPINET_INFO_VIRTUAL_ROOT_LIST;
```

cEntries: The number of entries contained in the list.

aVirtRootEntry: An array of **INET_INFO_VIRTUAL_ROOT_ENTRY** entries.

2.2.8 INET_INFO_VIRTUAL_ROOT_ENTRY

The **INET_INFO_VIRTUAL_ROOT_ENTRY** structure contains data describing a virtual root for the Internet protocol server instance.

This type is declared as follows:

```
typedef struct _INET_INFO_VIRTUAL_ROOT_ENTRY {
  [string] LPWSTR pszRoot;
  [string] LPWSTR pszAddress;
  [string] LPWSTR pszDirectory;
  DWORD dwMask;
  [string] LPWSTR pszAccountName;
  WCHAR AccountPassword[257];
  DWORD dwError;
} INET_INFO_VIRTUAL_ROOT_ENTRY,
 *LPINET_INFO_VIRTUAL_ROOT_ENTRY;
```

pszRoot: The virtual root name.

pszAddress: The optional IP address.

pszDirectory: The physical directory.

dwMask: The Access Mask for this virtual root.

pszAccountName: The account to connect as.

AccountPassword: Null-terminated WCHAR array containing the password for pszAccountName.<5>

dwError: The error code stored for the virtual root.

2.2.9 INET_INFO_SITE_LIST

The **INET_INFO_SITE_LIST** structure contains a list of INET_INFO_SITE_ENTRY site entries.

This type is declared as follows:

```
typedef struct _INET_INFO_SITE_LIST {
  DWORD cEntries;
  [size_is(cEntries)] INET_INFO_SITE_ENTRY aSiteEntry[];
} INET_INFO_SITE_LIST,
```

```
 *LPINET_INFO_SITE_LIST;
```

cEntries: The number of entries contained in the list.

aSiteEntry: An array of **INET_INFO_SITE_ENTRY** site entries.

2.2.10 INET_INFO_SITE_ENTRY

The **INET_INFO_SITE_ENTRY** structure contains information describing an Internet protocol server instance.

This type is declared as follows:

```
typedef struct _INET_INFO_SITE_ENTRY {
  [string] LPWSTR pszComment;
  DWORD dwInstance;
} INET_INFO_SITE_ENTRY,
 *LPINET_INFO_SITE_ENTRY;
```

pszComment: The server instance comment.

dwInstance: The server instance identifier.

2.2.11 INET_INFO_GLOBAL_CONFIG_INFO

The **INET_INFO_GLOBAL_CONFIG_INFO** structure contains configuration data global to all the Internet protocol services managed by this protocol.

This type is declared as follows:

```
typedef struct _INET_INFO_GLOBAL_CONFIG_INFO {
  DWORD FieldControl;
  DWORD BandwidthLevel;
  DWORD cbMemoryCacheSize;
} INET_INFO_GLOBAL_CONFIG_INFO,
 *LPINET_INFO_GLOBAL_CONFIG_INFO;
```

FieldControl: A bit-field that specifies the values of the **INET_INFO_GLOBAL_CONFIG_INFO** structure that have been initialized. An implementation MUST set the flag corresponding to the structure field when returning or updating configuration data. This field MUST be set to a valid combination of the following values.

Value	Meaning
0x00000001	FC_GINET_INFO_BANDWIDTH_LEVEL
0x00000002	FC_GINET_INFO_MEMORY_CACHE_SIZE

BandwidthLevel: The bytes per second to send over the network for the Internet protocol servers.

cbMemoryCacheSize: The size of the in-memory file cache for the Internet protocol servers.

2.2.12 INET_INFO_STATISTICS_INFO

The **INET_INFO_STATISTICS_INFO** union contains a pointer to an INET_INFO_STATISTICS_0 structure.

This type is declared as follows:

```
typedef
[switch_type(unsigned long)]
union _INET_INFO_STATISTICS_INFO {
  [case(0)]
    LPINET_INFO_STATISTICS_0 InetStats0;
  [default]     ;
} INET_INFO_STATISTICS_INFO,
 *LPINET_INFO_STATISTICS_INFO;
```

InetStats0: The pointer to an **INET_INFO_STATISTICS_0** structure that contains statistical information relevant to the Internet protocol server.

2.2.13 INET_INFO_STATISTICS_0

The **INET_INFO_STATISTICS_0** structure contains statistics for an Internet protocol server.

This type is declared as follows:

```
typedef struct _INET_INFO_STATISTICS_0 {
  INETA_CACHE_STATISTICS CacheCtrs;
  INETA_ATQ_STATISTICS AtqCtrs;
  DWORD nAuxCounters;
  DWORD rgCounters[20];
} INET_INFO_STATISTICS_0,
 *LPINET_INFO_STATISTICS_0;
```

CacheCtrs: The structure of type INETA_CACHE_STATISTICS that contains statistics on the Internet protocol server cache.

AtqCtrs: The structure of type INETA_ATQ_STATISTICS that contains statistics on the Internet protocol server network I/O.

nAuxCounters: The number of initialized elements in **rgCounters**. MUST be 0.

rgCounters: An array of 20 elements. This field is unused and MUST be ignored by clients.

2.2.14 INETA_ATQ_STATISTICS

The **INETA_ATQ_STATISTICS** structure contains network I/O statistics and client connection information for an Internet protocol server.

This type is declared as follows:

```
typedef struct _INETA_ATQ_STATISTICS {
  DWORD TotalBlockedRequests;
  DWORD TotalRejectedRequests;
  DWORD TotalAllowedRequests;
  DWORD CurrentBlockedRequests;
  DWORD MeasuredBandwidth;
} INETA_ATQ_STATISTICS,
 *LPINETA_ATQ_STATISTICS;
```

TotalBlockedRequests: The total number of blocked requests.

TotalRejectedRequests: The total number of rejected requests.

TotalAllowedRequests: The total number of allowed requests.

CurrentBlockedRequests: The total number of currently blocked requests.

MeasuredBandwidth: The measured network bandwidth in bytes per second.

2.2.15 INETA_CACHE_STATISTICS

The **INETA_CACHE_STATISTICS** structure contains statistics for the Internet protocol server's caches.<6>

This type is declared as follows:

```
typedef struct _INETA_CACHE_STATISTICS {
  DWORD FilesCached;
  DWORD TotalFilesCached;
  DWORD FileHits;
  DWORD FileMisses;
  DWORD FileFlushes;
  DWORDLONG CurrentFileCacheSize;
  DWORDLONG MaximumFileCacheSize;
  DWORD FlushedEntries;
  DWORD TotalFlushed;
  DWORD URICached;
  DWORD TotalURICached;
  DWORD URIHits;
  DWORD URIMisses;
  DWORD URIFlushes;
  DWORD TotalURIFlushed;
  DWORD BlobCached;
  DWORD TotalBlobCached;
  DWORD BlobHits;
  DWORD BlobMisses;
  DWORD BlobFlushes;
  DWORD TotalBlobFlushed;
} INETA_CACHE_STATISTICS,
 *LPINETA_CACHE_STATISTICS;
```

FilesCached: The current number of files whose content is in the Internet protocol server file cache.

TotalFilesCached: The total number of files whose content has been cached since Internet protocol server startup.

FileHits: The number of successful lookups in the Internet protocol server's file cache.

FileMisses: The number of unsuccessful lookups in the Internet protocol server's file cache.

FileFlushes: The number of file cache flushes since Internet protocol server startup.

CurrentFileCacheSize: The current number of bytes used for the Internet protocol server's file cache.

MaximumFileCacheSize: The maximum number of bytes used for the Internet protocol server's file cache.

FlushedEntries: The number of files that are marked for removal from the Internet protocol server cache after the current transfers are complete.

TotalFlushed: The number of files removed from the cache since Internet protocol server startup.

URICached: The number of URI information blocks currently cached by the Internet protocol server.

TotalURICached: The total number of URI information blocks ever added to the cache for the Internet protocol server.

URIHits: The number of successful lookups in the Internet protocol server's URI cache.

URIMisses: The number of unsuccessful lookups in the Internet protocol server's URI cache.

URIFlushes: The number of URI cache flushes since Internet protocol server startup.

TotalURIFlushed: The total number of URI information blocks that have been removed from the cache since Internet protocol server startup.

BlobCached: The number of BLOB information blocks currently cached by the Internet protocol server.

TotalBlobCached: The total number of BLOB information blocks ever added to the Internet protocol server's cache.

BlobHits: The number of successful lookups in the Internet protocol server's BLOB cache.

BlobMisses: The number of unsuccessful lookups in the Internet protocol server's BLOB cache.

BlobFlushes: The number of BLOB cache flushes since Internet protocol server startup.

TotalBlobFlushed: The total number of BLOB information blocks that have been removed from the cache since Internet protocol server startup.

2.2.16 INET_INFO_CAPABILITIES_STRUCT

The **INET_INFO_CAPABILITIES_STRUCT** structure specifies the features supported by an Internet protocol server implementation.

This type is declared as follows:

```
typedef struct _INET_INFO_CAPABILITIES_STRUCT {
  DWORD CapVersion;
  DWORD ProductType;
  DWORD MajorVersion;
  DWORD MinorVersion;
  DWORD BuildNumber;
  DWORD NumCapFlags;
  [size_is(NumCapFlags)] LPINET_INFO_CAP_FLAGS CapFlags;
} INET_INFO_CAPABILITIES_STRUCT,
 *LPINET_INFO_CAPABILITIES_STRUCT;
```

CapVersion: The version number of this structure. MUST be 1.

ProductType: The value that indicates the Windows operating system product type hosting the implementation. This field MUST be set to one of the following values.

Value	Meaning
INET_INFO_PRODUCT_NTSERVER 0x00000001	The operating system product type is a Windows server.
INET_INFO_PRODUCT_NTWKSTA 0x00000002	The operating system product type is a Windows client or Windows NT Workstation operating system.
INET_INFO_PRODUCT_UNKNOWN 0xFFFFFFFF	The operating system product type is unknown.

Value	Meaning
INET_INFO_PRODUCT_WINDOWS95 0x00000003	The operating system product type is Windows 95 operating system.

MajorVersion: The major version number of the Internet Information Services (IIS) Inetinfo Remote Protocol server.

MinorVersion: The minor version number of the Internet Information Services (IIS) Inetinfo Remote Protocol server.

BuildNumber: The build number of the version of the Windows operating system running on the Internet Information Services (IIS) Inetinfo Remote Protocol server.

NumCapFlags: The number of INET_INFO_CAP_FLAGS structures pointed to by the CapFlags member. MUST be 1.

CapFlags: The pointer to an array of **INET_INFO_CAP_FLAGS** structures that defines the server's capabilities.

2.2.17 INET_INFO_CAP_FLAGS

The **INET_INFO_CAP_FLAGS** structure contains information on features that are available for a protocol server.

This type is declared as follows:

```
typedef struct _INET_INFO_CAP_FLAGS {
  DWORD Flag;
  DWORD Mask;
} INET_INFO_CAP_FLAGS,
 *LPINET_INFO_CAP_FLAGS;
```

Flag: A value that indicates the features supported by the current running instance of the Internet protocol server implementation. The possible values for this member result from a bitwise OR of zero or more of the flags defined in the following table.

Value	Meaning
IIS_CAP1_ODBC_LOGGING 0x00000001	The Internet protocol server supports the Open Database Connectivity (ODBC) log format feature.
IIS_CAP1_FILE_LOGGING 0x00000002	The Internet protocol server supports the file system logging feature.
IIS_CAP1_VIRTUAL_SERVER 0x00000004	The Internet protocol server supports multiple instances of the protocol server network endpoint (1) aliases.
IIS_CAP1_BW_THROTTLING 0x00000008	The Internet protocol server supports network bandwidth throttling.
IIS_CAP1_IP_ACCESS_CHECK 0x00000010	The Internet protocol server supports blocking client connections using IP restrictions.
IIS_CAP1_MAX_CONNECTIONS 0x00000020	The Internet protocol server supports connection limiting.

Value	Meaning
IIS_CAP1_10_CONNECTION_LIMIT 0x00000040	The Internet protocol server supports a limit of 10 concurrent connections.
IIS_CAP1_MULTIPLE_INSTANCE 0x00000080	The Internet protocol server supports multiple instances.
IIS_CAP1_SSL_SUPPORT 0x00000100	The Internet protocol server supports the SSL protocol.
IIS_CAP1_OPERATORS_LIST 0x00000200	The Internet protocol server supports administrative operations by identities other than Windows operating system administrators.
IIS_CAP1_FP_INSTALLED 0x00000400	Front Page Server Extensions are installed on the server.
IIS_CAP1_CPU_AUDITING 0x00000800	The Internet protocol server supports CPU limits.
IIS_CAP1_SERVER_COMPRESSION 0x00001000	The Internet protocol server supports compression of network data.
IIS_CAP1_DAV 0x00002000	The Internet protocol server supports the WebDAV protocol.
IIS_CAP1_DIGEST_SUPPORT 0x00004000	The Internet protocol server supports the Digest Authentication Protocol.
IIS_CAP1_NT_CERTMAP_SUPPORT 0x00008000	The Internet protocol server supports mapping client certificates to Windows user accounts.
IIS_CAP1_POOLED_OOP 0x00010000	The Internet protocol server supports running a pool of applications in a separate process.

Mask: A value that indicates the capabilities that can be enabled for the protocol servers in the current implementation. The possible values for this member result from a bitwise OR operation of zero or more of the following flags.

Note The **Mask** value contains all the capabilities that the current version of the protocol server implementation can support. The **Flag** value indicates the features that the current running server instance does support.The server sets the mask value to a bitwise OR of all the flags it knows about. The server then sets the flags to the bitwise OR of the features supported for the current platform, a subset of those set in the mask field. A given version of the software reports the same mask values, but might support different flags values depending on the operating system type.

Value	Meaning
IIS_CAP1_ODBC_LOGGING 0x00000001	The Internet protocol server supports the Open Database Connectivity (ODBC) log format feature.
IIS_CAP1_FILE_LOGGING 0x00000002	The Internet protocol server supports the file system logging feature.
IIS_CAP1_VIRTUAL_SERVER 0x00000004	The Internet protocol server supports multiple instances of the protocol server network endpoint (2) aliases.
IIS_CAP1_BW_THROTTLING	The Internet protocol server supports network bandwidth throttling.

Value	Meaning
0x00000008	
IIS_CAP1_IP_ACCESS_CHECK 0x00000010	The Internet protocol server supports blocking client connections by using IP restrictions.
IIS_CAP1_MAX_CONNECTIONS 0x00000020	The Internet protocol server supports connection limiting.
IIS_CAP1_10_CONNECTION_LIMIT 0x00000040	The Internet protocol server supports a limit of 10 concurrent connections.
IIS_CAP1_MULTIPLE_INSTANCE 0x00000080	The Internet protocol server supports multiple instances.
IIS_CAP1_SSL_SUPPORT 0x00000100	The Internet protocol server supports the SSL protocol.
IIS_CAP1_OPERATORS_LIST 0x00000200	The Internet protocol server supports administrative operations by identities other than Windows operating system administrators.
IIS_CAP1_FP_INSTALLED 0x00000400	Front Page Server Extensions are installed on the server.
IIS_CAP1_CPU_AUDITING 0x00000800	The Internet protocol server supports CPU limits.
IIS_CAP1_SERVER_COMPRESSION 0x00001000	The Internet protocol server supports compression of network data.
IIS_CAP1_DAV 0x00002000	The Internet protocol server supports the WebDAV protocol.
IIS_CAP1_DIGEST_SUPPORT 0x00004000	The Internet protocol server supports the Digest Authentication Protocol.
IIS_CAP1_NT_CERTMAP_SUPPORT 0x00008000	The Internet protocol server supports mapping client certificates to Windows user accounts.
IIS_CAP1_POOLED_OOP 0x00010000	The Internet protocol server supports running a pool of applications in a separate process.

2.2.18 W3_STATISTICS_STRUCT

The **W3_STATISTICS_STRUCT** union contains a pointer to a W3_STATISTICS_1 structure.

This type is declared as follows:

```
typedef
[switch_type(unsigned long)]
union _W3_STATISTICS_UNION {
  [case(0)]
    LPW3_STATISTICS_1 StatInfo1;
  [default]     ;
} W3_STATISTICS_STRUCT,
 *LPW3_STATISTICS_STRUCT;
```

StatInfo1: The pointer to a **W3_STATISTICS_1** structure that contains the HTTP protocol statistics.

2.2.19 W3_STATISTICS_1

The **W3_STATISTICS_1** structure contains statistics on the usage of the HTTP server.

This type is declared as follows:

```
typedef struct _W3_STATISTICS_1 {
  LARGE_INTEGER TotalBytesSent;
  LARGE_INTEGER TotalBytesReceived;
  DWORD TotalFilesSent;
  DWORD TotalFilesReceived;
  DWORD CurrentAnonymousUsers;
  DWORD CurrentNonAnonymousUsers;
  DWORD TotalAnonymousUsers;
  DWORD TotalNonAnonymousUsers;
  DWORD MaxAnonymousUsers;
  DWORD MaxNonAnonymousUsers;
  DWORD CurrentConnections;
  DWORD MaxConnections;
  DWORD ConnectionAttempts;
  DWORD LogonAttempts;
  DWORD TotalOptions;
  DWORD TotalGets;
  DWORD TotalPosts;
  DWORD TotalHeads;
  DWORD TotalPuts;
  DWORD TotalDeletes;
  DWORD TotalTraces;
  DWORD TotalMove;
  DWORD TotalCopy;
  DWORD TotalMkcol;
  DWORD TotalPropfind;
  DWORD TotalProppatch;
  DWORD TotalSearch;
  DWORD TotalLock;
  DWORD TotalUnlock;
  DWORD TotalOthers;
  DWORD TotalCGIRequests;
  DWORD TotalBGIRequests;
  DWORD TotalNotFoundErrors;
  DWORD TotalLockedErrors;
  DWORD CurrentCalAuth;
  DWORD MaxCalAuth;
  DWORD TotalFailedCalAuth;
  DWORD CurrentCalSsl;
  DWORD MaxCalSsl;
  DWORD TotalFailedCalSsl;
  DWORD CurrentCGIRequests;
  DWORD CurrentBGIRequests;
  DWORD MaxCGIRequests;
  DWORD MaxBGIRequests;
  DWORD CurrentBlockedRequests;
  DWORD TotalBlockedRequests;
  DWORD TotalAllowedRequests;
  DWORD TotalRejectedRequests;
  DWORD MeasuredBw;
  DWORD ServiceUptime;
  DWORD TimeOfLastClear;
  DWORD nAuxCounters;
  DWORD rgCounters[20];
} W3_STATISTICS_1,
 *LPW3_STATISTICS_1;
```

TotalBytesSent: The total number of bytes sent.

TotalBytesReceived: The total number of bytes received.

TotalFilesSent: The total number of files sent by the HTTP server.

TotalFilesReceived: The total number of files received by the HTTP server.

CurrentAnonymousUsers: The current number of anonymous users connected to the HTTP server.

CurrentNonAnonymousUsers: The current number of non-anonymous users connected to the HTTP server.

TotalAnonymousUsers: The total number of anonymous users that have ever connected to the HTTP server.

TotalNonAnonymousUsers: The total number of non-anonymous users that have ever connected to the HTTP server.

MaxAnonymousUsers: The maximum number of anonymous users who simultaneously connected to the HTTP server.

MaxNonAnonymousUsers: The maximum number of non-anonymous users who simultaneously connected to the HTTP server.

CurrentConnections: The current number of connections to the HTTP server.

MaxConnections: The maximum number of connections to the HTTP server.

ConnectionAttempts: The number of connection attempts that have been made to the HTTP server.

LogonAttempts: The number of logon attempts that have been made to the HTTP server.

TotalOptions: The total number of HTTP requests made with the OPTIONS method.

TotalGets: The total number of HTTP requests made using the GET method.

TotalPosts: The total number of HTTP requests made using the POST method.

TotalHeads: The total number of HTTP requests made using the HEAD method.

TotalPuts: The total number of HTTP requests made using the PUT method.

TotalDeletes: The total number of HTTP requests made using the DELETE method.

TotalTraces: The total number of HTTP requests made using the TRACE method.

TotalMove: The total number of WebDAV requests made using the MOVE method. For more information on WebDAV requests, see [RFC2518].

TotalCopy: The total number of WebDAV requests made using the COPY method.

TotalMkcol: The total number of WebDAV requests made using the MKCOL method.

TotalPropfind: The total number of WebDAV requests made using the PROPFIND method.

TotalProppatch: The total number of WebDAV requests made using the PROPPATCH method.

TotalSearch: The total number of requests made using the SEARCH method.

TotalLock: The total number of WebDAV requests made using the LOCK method.

TotalUnlock: The total number of WebDAV requests made using the UNLOCK method.

TotalOthers: The total number of HTTP requests made to methods not already listed.

TotalCGIRequests: The total number of Common Gateway Interface (CGI) requests ever made to the HTTP server.

TotalBGIRequests: The total number of Binary Gateway Interface (BGI) requests ever made to the HTTP server.

TotalNotFoundErrors: The total number of requests that could not be satisfied by the server because the requested document could not be found. These requests are generally reported as an HTTP 404 error code to the client.

TotalLockedErrors: The total number of locked errors.

CurrentCalAuth: The current number of Client Access Licenses (CALs) that are authorized (for more information, see [MSFT-CAL]).

MaxCalAuth: The maximum number of CALs that are authorized.

TotalFailedCalAuth: The total number of failed CAL authorizations.

CurrentCalSsl: The current number of CALs for a Secure Sockets Layer (SSL) connection.

MaxCalSsl: The maximum number of CALs for an SSL connection.

TotalFailedCalSsl: The total number of failed CAL SSL connections.

CurrentCGIRequests: The current number of CGI requests. For more information on CGI, see [RFC3875].

CurrentBGIRequests: The current number of BGI requests.

MaxCGIRequests: The maximum number of CGI requests allowed.

MaxBGIRequests: The maximum number of BGI requests allowed.

CurrentBlockedRequests: The current number of blocked requests.

TotalBlockedRequests: The total number of blocked requests.

TotalAllowedRequests: The total number of allowed requests to the HTTP server.

TotalRejectedRequests: The total number of rejected requests.

MeasuredBw: The measured network bandwidth for the HTTP server.

ServiceUptime: The HTTP server uptime.

TimeOfLastClear: The time of the last clear.

nAuxCounters: The number of initialized elements in **rgCounters**. MUST be 0.

rgCounters: An array of 20 elements. This field is unused and MUST be ignored by clients.

2.2.20 FTP_STATISTICS_STRUCT

The **FTP_STATISTICS_STRUCT** union contains a pointer to an FTP_STATISTICS_0 structure.

This type is declared as follows:

```
typedef
[switch_type(unsigned long)]
union _FTP_STATISTICS_UNION {
  [case(0)]
```

```
    LPFTP_STATISTICS_0 StatInfo0;
  [default]     ;
} FTP_STATISTICS_STRUCT,
 *LPFTP_STATISTICS_STRUCT;
```

StatInfo0: The pointer to an **FTP_STATISTICS_0** structure that contains the FTP server statistics.

2.2.21 FTP_STATISTICS_0

The **FTP_STATISTICS_0** structure contains statistics on the usage of the FTP server.

This type is declared as follows:

```
typedef struct _FTP_STATISTICS_0 {
  LARGE_INTEGER TotalBytesSent;
  LARGE_INTEGER TotalBytesReceived;
  DWORD TotalFilesSent;
  DWORD TotalFilesReceived;
  DWORD CurrentAnonymousUsers;
  DWORD CurrentNonAnonymousUsers;
  DWORD TotalAnonymousUsers;
  DWORD TotalNonAnonymousUsers;
  DWORD MaxAnonymousUsers;
  DWORD MaxNonAnonymousUsers;
  DWORD CurrentConnections;
  DWORD MaxConnections;
  DWORD ConnectionAttempts;
  DWORD LogonAttempts;
  DWORD ServiceUptime;
  DWORD TotalAllowedRequests;
  DWORD TotalRejectedRequests;
  DWORD TotalBlockedRequests;
  DWORD CurrentBlockedRequests;
  DWORD MeasuredBandwidth;
  DWORD TimeOfLastClear;
} FTP_STATISTICS_0,
 *LPFTP_STATISTICS_0;
```

TotalBytesSent: The total number of bytes sent.

TotalBytesReceived: The total number of bytes received.

TotalFilesSent: The total number of files sent by the FTP server.

TotalFilesReceived: The total number of files received by the FTP server.

CurrentAnonymousUsers: The current number of anonymous users connected to the FTP server.

CurrentNonAnonymousUsers: The current number of non-anonymous users connected to the FTP server.

TotalAnonymousUsers: The total number of anonymous users that have ever connected to the FTP server.

TotalNonAnonymousUsers: The total number of non-anonymous users that have ever connected to the FTP server.

MaxAnonymousUsers: The maximum number of anonymous users allowed to simultaneously connect to the FTP server.

MaxNonAnonymousUsers: The maximum number of non-anonymous users allowed to simultaneously connect to the FTP server.

CurrentConnections: The current number of connections to the FTP server.

MaxConnections: The maximum number of connections to the FTP server.

ConnectionAttempts: The number of connection attempts that have been made to the FTP server.

LogonAttempts: The number of logon attempts that have been made to the FTP server.

ServiceUptime: The time that the FTP server has been operational.

TotalAllowedRequests: The total number of requests allowed to the FTP server.

TotalRejectedRequests: The total number of rejected requests.

TotalBlockedRequests: The total number of blocked requests.

CurrentBlockedRequests: The current number of blocked requests.

MeasuredBandwidth: The measured network bandwidth for the FTP server.

TimeOfLastClear: The time of the last clear.

2.2.22 IIS_USER_ENUM_STRUCT

The **IIS_USER_ENUM_STRUCT** structure contains a pointer to an IIS_USER_INFO_1_CONTAINER.

This type is declared as follows:

```
typedef struct _IIS_USER_ENUM_STRUCT {
    DWORD Level;
    [switch_is(Level)] union _USER_ENUM_UNION {
        [case(1)]
            LPIIS_USER_INFO_1_CONTAINER Level1;
        [default]           ;
    } ConfigInfo;
} IIS_USER_ENUM_STRUCT,
 *LPIIS_USER_ENUM_STRUCT;
```

Level: The value that indicates the level of detail in the information provided. This member MUST be set to 1.

ConfigInfo: The name of the contained union.

Level1: The pointer to an **IIS_USER_INFO_1_CONTAINER** structure that contains the user information collection.

2.2.23 IIS_USER_INFO_1_CONTAINER

The **IIS_USER_INFO_1_CONTAINER** structure contains a list of IIS_USER_INFO_1 structures describing users who are actively connected to the Internet protocol server.

This type is declared as follows:

```
typedef struct _IIS_USER_INFO_1_CONTAINER {
  DWORD EntriesRead;
  [size_is(EntriesRead)] LPIIS_USER_INFO_1 Buffer;
} IIS_USER_INFO_1_CONTAINER,
 *LPIIS_USER_INFO_1_CONTAINER;
```

EntriesRead: The total number of IIS_USER_INFO objects in Buffer.

Buffer: The pointer to an array of **IIS_USER_INFO_1** structures.

2.2.24 IIS_USER_INFO_1

The **IIS_USER_INFO_1** structure describes a user connected to an Internet protocol server.

This type is declared as follows:

```
typedef struct _IIS_USER_INFO_1 {
  DWORD idUser;
  [string] LPWSTR pszUser;
  BOOL fAnonymous;
  DWORD inetHost;
  DWORD tConnect;
} IIS_USER_INFO_1,
 *LPIIS_USER_INFO_1;
```

idUser: A unique identifier for the user.

pszUser: A name for the user, not necessarily unique.

fAnonymous: Indicates whether or not the user connected anonymously. This field MUST be one of the following values.

Value	Meaning
TRUE 1	The user is logged on as Anonymous.
FALSE 0	The user is not logged on as Anonymous.

inetHost: The host IPv4 address. Data is stored in network byte order.

tConnect: The user connection time measured in elapsed seconds.

2.2.25 Common Error Codes

Unless specified explicitly, the methods of the Internet Information Services (IIS) Inetinfo Remote Protocol MUST return 0 to indicate success and a nonzero implementation-specific value to indicate failure in the return code of the response. All failure values MUST be treated as equivalent for protocol purposes and SHOULD simply be passed back to the invoking application. A list of error codes that are potentially returned is available, as specified in [MS-ERREF].<7>

3 Protocol Details

The following sections specify details of the Internet Information Services (IIS) Inetinfo Remote Protocol, including abstract data models, interface method syntax, and message processing rules.

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

3.1 Inetinfo Server Details

The IIS Inetinfo Remote Protocol server handles client requests for any of the messages that are specified in section 2. For each of those messages, the behavior of the server is specified in section 3.1.4.

An implementation MAY implement only a subset of the methods specified in the inetinfo interface. Or a particular Internet protocol server might only be able to support a subset of methods specified in the interface. For methods that are not implemented, the server SHOULD return ERROR_NOT_SUPPORTED (0x00000032).

3.1.1 Abstract Data Model

The Internet Information Services (IIS) Inetinfo Remote Protocol provides runtime status and statistical data and manages runtime configuration for the Internet protocol server. The Internet protocol server MUST provide a mechanism to collect runtime data and expose it through the Internet Information Services (IIS) Inetinfo Remote Protocol server. Some methods operate on configuration data that SHOULD be persisted beyond the lifetime of an Internet protocol server process.

3.1.2 Timers

None.

3.1.3 Initialization

The Internet Information Services (IIS) Inetinfo Remote Protocol server MUST be initialized by registering the RPC interface and listening on the RPC well-known endpoint, as specified in section 2.1. The server MUST then wait for Internet Information Services (IIS) Inetinfo Remote Protocol clients to establish a connection.

3.1.4 Higher-Layer Triggered Events

The Internet Information Services (IIS) Inetinfo Remote Protocol is invoked explicitly by an application.

3.1.5 Message Processing Events and Sequencing Rules

The remainder of this section describes the server behavior for the RPC methods on the inetinfo interface that makes up the IIS Inetinfo Remote Protocol. IIS Inetinfo Remote Protocol clients can invoke the RPC methods that are specified in this section in any order after an Internet Information Services (IIS) Inetinfo Remote Protocol session is established with the server. The outcome of the calls depends on the parameters that are passed to each of those calls and not on a particular call sequence or state maintained across method invocations.

Methods in RPC Opnum Order

Method	Description
R_InetInfoGetVersion	Called by the client. In response, the server returns its version information. Opnum: 0
R_InetInfoGetAdminInformation	Called by the client. In response, the server returns configuration data for the specified Internet protocol server. Opnum: 1
R_InetInfoGetSites	Called by the client. In response, the server retrieves a list of service instances for the specified Internet protocol server. Opnum: 2
R_InetInfoSetAdminInformation	Called by the client. In response, the server sets configurable properties for the specified Internet protocol server. Opnum: 3
R_InetInfoGetGlobalAdminInformation	Called by the client. In response, the server retrieves configuration data shared by all Internet protocol servers. Opnum: 4
R_InetInfoSetGlobalAdminInformation	Called by the client. In response, the server sets configuration data shared by all Internet protocol servers. Opnum: 5
R_InetInfoQueryStatistics	Called by the client. In response, the server retrieves statistical data for the specified Internet protocol server. Opnum: 6
R_InetInfoClearStatistics	Called by the client. In response, the server resets the statistical data maintained by the specified Internet protocol server. Opnum: 7
R_InetInfoFlushMemoryCache	Called by the client. In response, the server flushes data from the internal caches of the specified Internet protocol server. Opnum: 8
R_InetInfoGetServerCapabilities	Called by the client. In response, the server returns information on the features of the Internet protocol servers and the host operating system. Opnum: 9
R_W3QueryStatistics2	Called by the client. In response, the server returns statistical data from the HTTP server. Opnum: 10
R_W3ClearStatistics2	Called by the client. In response, the server resets statistical data for the HTTP server. Opnum: 11
R_FtpQueryStatistics2	Called by the client. In response, the server returns statistical data from the FTP server. Opnum: 12
R_FtpClearStatistics2	Called by the client. In response, the server resets statistical data for the FTP server. Opnum: 13
R_IISEnumerateUsers	Called by the client. In response, the server returns a list of clients connected to the specified Internet protocol server.

Method	Description
	Opnum: 14
R_IISDisconnectUser	Called by the client. In response, the server disconnects the specified user from the specified Internet protocol server. Opnum: 15
Opnum16NotUsedOnWire	Reserved for local use. Opnum: 16
Opnum17NotUsedOnWire	Reserved for local use. Opnum: 17

Structures

The **Message Processing Events and Sequencing Rules** interface defines the following structures.

Structure	Description
INET_INFO_CONFIG_INFO	This structure stores configuration values for an Internet protocol server.
INET_LOG_CONFIGURATION	This structure contains configuration information for Internet protocol server logging.
INET_INFO_IP_SEC_LIST	This structure contains a list of INET_INFO_IP_SEC_ENTRY entries.
INET_INFO_IP_SEC_ENTRY	This structure contains Internet protocol security (IPv4) entries.
INET_INFO_VIRTUAL_ROOT_LIST	This structure contains a list of INET_INFO_VIRTUAL_ROOT_ENTRY virtual root entries.
INET_INFO_VIRTUAL_ROOT_ENTRY	This structure contains data describing a virtual root for the Internet protocol server instance.
INET_INFO_SITE_LIST	This structure contains a list of INET_INFO_SITE_ENTRY site entries.
INET_INFO_SITE_ENTRY	This structure contains information describing an Internet protocol server instance.
INET_INFO_GLOBAL_CONFIG_INFO	This structure contains configuration data global to all the Internet protocol services managed by this protocol.
INET_INFO_STATISTICS_0	This structure contains statistics for an Internet protocol server.
INETA_ATQ_STATISTICS	This structure contains network I/O statistics and client connection information for an Internet protocol server.
INETA_CACHE_STATISTICS	This structure contains statistics for the Internet protocol server's caches.
INET_INFO_CAPABILITIES_STRUCT	This structure specifies the features supported by an Internet protocol server implementation.
INET_INFO_CAP_FLAGS	This structure contains information on features that are available for a protocol server.
W3_STATISTICS_1	This structure contains statistics on the usage of the HTTP server.
FTP_STATISTICS_0	This structure contains statistics on the usage of the FTP server.
IIS_USER_ENUM_STRUCT	This structure contains a pointer to an

Structure	Description
	IIS_USER_INFO_1_CONTAINER.
IIS_USER_INFO_1_CONTAINER	This structure contains a list of IIS_USER_INFO_1 structures describing users who are actively connected to the Internet protocol server.
IIS_USER_INFO_1	This structure describes a user connected to an Internet protocol server.

Unions

The **Message Processing Events and Sequencing Rules** interface defines the following unions.

Union	Description
INET_INFO_STATISTICS_INFO	This union contains a pointer to an **INET_INFO_STATISTICS_0** structure.
W3_STATISTICS_STRUCT	This union contains a pointer to a **W3_STATISTICS_1** structure.
FTP_STATISTICS_STRUCT	This union contains a pointer to an **FTP_STATISTICS_0** structure.

In the preceding tables, "Reserved for local use" means that the client MUST NOT send the opnum, and the server behavior is undefined because it does not affect interoperability.<8>

3.1.5.1 R_InetInfoGetVersion (Opnum 0)

The **R_InetInfoGetVersion** method is called by the client. In response, the server returns its version information.

```
DWORD R_InetInfoGetVersion(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwReserved,
  [out] DWORD* pdwVersion
);
```

pszServer: A custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwReserved: An unused parameter. MUST be ignored by the server implementation.

pdwVersion: A pointer to a variable. On successful return, it MUST contain a major and minor version number for the server implementation. The major version is stored in the low WORD, and the minor version is stored in the high WORD.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or [MS-ERREF] section 2.3.1.

ERROR_SUCCESS (0x00000000)

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

The value returned in *pdwVersion* SHOULD<9> correspond to the version of the Internet protocol servers managed by the Internet Information Services (IIS) Inetinfo Remote Protocol server.

3.1.5.2 R_InetInfoGetAdminInformation (Opnum 1)

The **R_InetInfoGetAdminInformation** method is called by the client. In response, the server retrieves configuration data for the specified Internet protocol server.

```
DWORD R_InetInfoGetAdminInformation(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwServerMask,
  [out] LPINET_INFO_CONFIG_INFO* ppConfig
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwServerMask: The identifier for the target Internet protocol server, as specified in section 2.2.2.

ppConfig: The pointer to a pointer to an INET_INFO_CONFIG_INFO structure that contains configuration data for the specified Internet protocol server.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or [MS-ERREF] section 2.3.1. The most common error codes are listed in the following table.

Return value/code	Description
0x00000002 ERROR_FILE_NOT_FOUND	The system cannot find the specified file.
0x00000426 ERROR_SERVICE_NOT_ACTIVE	The service is not running.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server MUST return the data specified in *ppConfig* and set the **FieldControl** member, as specified in **INET_INFO_CONFIG_INFO**.

Data returned MAY be a configuration that is persisted in a configuration store for the protocol server, runtime or derived data, or default operating values.

An implementation MAY support only a subset of the configuration data specified in the **INET_INFO_CONFIG_INFO** structure, but it MUST set the **FieldControl** member for any valid value returned.

3.1.5.3 R_InetInfoGetSites (Opnum 2)

The **R_InetInfoGetSites** method is called by the client. In response, the server retrieves a list of server instances for the specified Internet protocol server.

```
DWORD R_InetInfoGetSites(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwServerMask,
  [out] LPINET_INFO_SITE_LIST* ppSites
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwServerMask: The identifier for the target Internet protocol server, as specified in section 2.2.2.

ppSites: The pointer to a pointer to INET_INFO_SITE_LIST that specifies the list of defined server instances for the Internet protocol server specified by *dwServerMask*.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or [MS-ERREF] section 2.3.1. The most common error codes are listed in the following table.

Return value/code	Description
0x00000426 ERROR_SERVICE_NOT_ACTIVE	The service is not running.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server SHOULD return a list of defined server instances in *ppSites* if the return code indicates success.<10>

3.1.5.4 R_InetInfoSetAdminInformation (Opnum 3)

The **R_InetInfoSetAdminInformation** method is called by the client. In response, the server sets configurable properties for the specified Internet protocol server.

```
DWORD R_InetInfoSetAdminInformation(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwServerMask,
  [in, ref] INET_INFO_CONFIG_INFO* pConfig
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwServerMask: The identifier for the target Internet protocol server, as specified in section 2.2.2.

pConfig: The pointer to an INET_INFO_CONFIG_INFO structure containing the property configuration to set. The client MUST set the appropriate flag in the **FieldControl** member for any data field in pConfig that is to be set by the server.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2, or [MS-ERREF] section 2.3.1. The most common error codes are listed in the following table.

Return value/code	Description
0x00000002 ERROR_FILE_NOT_FOUND	The system cannot find the specified file.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000426 ERROR_SERVICE_NOT_ACTIVE	The service is not running.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server SHOULD set the configurable properties in *pConfig* into the configuration store for the Internet protocol server specified by *dwServerMask*.

The server MUST NOT access a field value in *pConfig* unless the corresponding flag in the **FieldControl** field is set.

The implementation MAY choose not to save some fields of the **INET_INFO_CONFIG_INFO** structure. If so, the server MUST ignore the field values sent by the client.

The implementation MAY return an error if it is unable to persist a field value due to some internal error.

3.1.5.5 R_InetInfoGetGlobalAdminInformation (Opnum 4)

The **R_InetInfoGetGlobalAdminInformation** method is called by the client. In response, the server retrieves configuration data shared by all Internet protocol servers.

```
DWORD R_InetInfoGetGlobalAdminInformation(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwServerMask,
  [out] LPINET_INFO_GLOBAL_CONFIG_INFO* ppConfig
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwServerMask: This value MUST be ignored by the server.

ppConfig: The pointer to a pointer to an INET_INFO_GLOBAL_CONFIG_INFO structure.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or [MS-ERREF] section 2.3.1.

ERROR_SUCCESS (0x00000000)

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server MUST return the configuration data in *ppConfig*, if the return code indicates success.

3.1.5.6 R_InetInfoSetGlobalAdminInformation (Opnum 5)

The **R_InetInfoSetGlobalAdminInformation** assigns global settings for all Internet protocol servers present on the host system.

```
DWORD R_InetInfoSetGlobalAdminInformation(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwServerMask,
  [in, ref] INET_INFO_GLOBAL_CONFIG_INFO* pConfig
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwServerMask: This value MUST be ignored by the server.

pConfig: The pointer to an INET_INFO_GLOBAL_CONFIG_INFO structure that contains global administrative information.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or [MS-ERREF] section 2.3.1. The most common error codes are listed in the following table.

Return value/code	Description
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000032 ERROR_NOT_SUPPORTED	The request is not supported.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server MUST set the configurable properties in *pConfig* into the configuration store for the Internet protocol servers if the return code indicates success.

3.1.5.7 R_InetInfoQueryStatistics (Opnum 6)

The **R_InetInfoQueryStatistics** method is called by the client. In response, the server retrieves statistical data for the specified Internet protocol server.

```
DWORD R_InetInfoQueryStatistics(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD Level,
  [in] DWORD dwServerMask,
  [out, switch_is(Level)] LPINET_INFO_STATISTICS_INFO StatsInfo
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

Level: The level of detail to be retrieved. This member MUST be set to 0. If another value is sent by the client, the server MUST return ERROR_INVALID_LEVEL (0x0000007C).

dwServerMask: The identifier for the target Internet protocol server, as specified in section 2.2.2. A value of 0 indicates that aggregate statistical data is to be returned for all protocol servers.

StatsInfo: The pointer to an INET_INFO_STATISTICS_INFO union that contains the data to be returned.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or [MS-ERREF] section 2.3.1. The following table describes possible error code values.

Return value/code	Description
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server MAY return the statistical data in *StatsInfo* if the return code indicates success.<11>

3.1.5.8 R_InetInfoClearStatistics (Opnum 7)

The **R_InetInfoClearStatistics** is called by the client. In response, the server resets the statistical data maintained by the specified Internet protocol server.

```
DWORD R_InetInfoClearStatistics(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwServerMask
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwServerMask: The identifier for the target Internet protocol server, as specified in section 2.2.2.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or [MS-ERREF] section 2.3.1. The most common error codes are listed in the following table.

Return value/code	Description
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000032 ERROR_NOT_SUPPORTED	The request is not supported.
0x00000057 ERROR_INVALID_PARAMETER	The parameter is incorrect.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server MAY clear any accumulated data that would be returned by R_InetInfoQueryStatistics.<12>

3.1.5.9 R_InetInfoFlushMemoryCache (Opnum 8)

The **R_InetInfoFlushMemoryCache** method is called by the client. In response, the server flushes data from the internal caches of the specified Internet protocol server.

```
DWORD R_InetInfoFlushMemoryCache(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwServerMask
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwServerMask: The identifier for the target Internet protocol server, as specified in section 2.2.2.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or [MS-ERREF] section 2.3.1. The most common error codes are listed in the following table.

Return value/code	Description
0x00000005 ERROR_ACCESS_DENIED	Access is denied.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server MUST flush data from the internal caches of the specified Internet protocol server. If *dwServerMask* is 0, caches are flushed for all protocol servers.

3.1.5.10 R_InetInfoGetServerCapabilities (Opnum 9)

The **R_InetInfoGetServerCapabilities** method is called by the client. In response, the server returns information on the features of the Internet protocol servers and the host operating system.

```
DWORD R_InetInfoGetServerCapabilities(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwReserved,
  [out] LPINET_INFO_CAPABILITIES_STRUCT* ppCap
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwReserved: This value MUST be ignored by the server.

ppCap: The pointer to an INET_INFO_CAPABILITIES_STRUCT structure that indicates the capabilities of the Internet protocol servers on the host system.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or [MS-ERREF] section 2.3.1.

ERROR_SUCCESS (0x00000000)

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server MUST initialize *ppCap* with those features supported by the current version of the Internet protocol servers if the return code indicates success.

3.1.5.11 R_W3QueryStatistics2 (Opnum 10)

The **R_W3QueryStatistics2** method is called by the client. In response, the server returns statistical data from the HTTP server.

```
DWORD R_W3QueryStatistics2(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwLevel,
  [in] DWORD dwInstance,
  [in] DWORD dwReserved,
  [out, switch_is(dwLevel)] LPW3_STATISTICS_STRUCT InfoStruct
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwLevel: The level of detail to be retrieved. This parameter MUST be set to 0 by the client. Other values MUST generate a return code of ERROR_INVALID_LEVEL (0x0000007C).

dwInstance: The ID of the protocol server instance whose statistical data is being requested. The following values have special meanings.

Value	Meaning
0x000000000	Return global (not per server instance) statistics data.
0xf0000003	Return statistics aggregated across all protocol server instances.

dwReserved: This value MUST be ignored by the server.

InfoStruct: The pointer to a W3_STATISTICS_STRUCT union to contain the retrieved statistics data.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or [MS-ERREF] section 2.3.1. The most common error codes are listed in the following table.

Return value/code	Description
0x00000002 ERROR_FILE_NOT_FOUND	The system cannot find the specified file.
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x00000426 ERROR_SERVICE_NOT_ACTIVE	The service is not running.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server MUST initialize *InfoStruct* with the statistical data for the HTTP server if the return code indicates success.

3.1.5.12 R_W3ClearStatistics2 (Opnum 11)

The **R_W3ClearStatistics2** method is called by the client. In response, the server resets statistical data for the HTTP server.

```
DWORD R_W3ClearStatistics2(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwInstance
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. Value MUST NOT be used by the server implementation.

dwInstance: The ID of the protocol server instance whose statistical data is being cleared.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or [MS-ERREF] section 2.3.1. The most common error codes are listed in the following table.

Return value/code	Description
0x00000002 ERROR_FILE_NOT_FOUND	The system cannot find the specified file.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000426 ERROR_SERVICE_NOT_ACTIVE	The service is not running.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server MAY reset the statistical data for the HTTP server if the return code indicates success. If the data is reset, a time stamp SHOULD be saved. This time stamp SHOULD be used to populate the **TimeOfLastClear** field of the W3_STATISTICS_1 structure in subsequent calls to R_W3QueryStatistics2.<13>

3.1.5.13 R_FtpQueryStatistics2 (Opnum 12)

The **R_FtpQueryStatistics2** method is called by the client. In response, the server returns statistical data from the FTP server.

```
DWORD R_FtpQueryStatistics2(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwLevel,
  [in] DWORD dwInstance,
  [in] DWORD dwReserved,
  [out, switch_is(dwLevel)] LPFTP_STATISTICS_STRUCT InfoStruct
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwLevel: The level of detail to be retrieved. This parameter MUST be set to 0 by the client. Other values MUST generate a return code of ERROR_INVALID_LEVEL (0x0000007C).

dwInstance: The ID of the protocol server instance whose statistical data is being requested. The following values have special meanings.

Value	Meaning
0x00000000	Return global (not per server instance) statistics data.
0xF0000003	Return statistics aggregated across all protocol server instances.

dwReserved: This value MUST be ignored by the server.

InfoStruct: The pointer to an FTP_STATISTICS_STRUCT union to contain the retrieved statistics data.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or [MS-ERREF] section 2.3.1. The most common error codes are listed in the following table.

Return value/code	Description
0x00000002	The system cannot find the specified file.

Return value/code	Description
ERROR_FILE_NOT_FOUND	
0x0000007C ERROR_INVALID_LEVEL	The system call level is not correct.
0x00000426 ERROR_SERVICE_NOT_ACTIVE	The service is not running.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server MUST initialize *InfoStruct* with the statistical data for the FTP server if the return code indicates success.

3.1.5.14 R_FtpClearStatistics2 (Opnum 13)

The **R_FtpClearStatistics2** method is called by the client. In response, the server resets statistical data for the FTP server.

```
DWORD R_FtpClearStatistics2(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwInstance
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwInstance: The ID of the protocol server instance whose statistical data is being cleared.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or [MS-ERREF] section 2.3.1. The most common error codes are listed in the following table.

Return value/code	Description
0x00000002 ERROR_FILE_NOT_FOUND	The system cannot find the specified file.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000426 ERROR_SERVICE_NOT_ACTIVE	The service is not running.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server MAY reset the statistical data for the FTP server if the return code indicates success. If the data is reset, a time stamp SHOULD be saved. This time stamp SHOULD be used to populate the **TimeOfLastClear** field of the FTP_STATISTICS_0 structure in subsequent calls to R_FtpQueryStatistics2.<14>

3.1.5.15 R_IISEnumerateUsers (Opnum 14)

The **R_IISEnumerateUsers** method is called by the client. In response, the server returns a list of clients connected to the specified Internet protocol server.

```
DWORD R_IISEnumerateUsers(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwServiceId,
  [in] DWORD dwInstance,
  [in, out] LPIIS_USER_ENUM_STRUCT InfoStruct
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwServiceId: The identifier for the specified Internet protocol server, as specified in section 2.2.2.

dwInstance: The ID of the Internet protocol server instance whose users are being enumerated.

InfoStruct: The pointer to an IIS_USER_ENUM_STRUCT that contains the list of active users for this server.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or[MS-ERREF] section 2.3.1. The most common error codes are listed in the following table.

Return value/code	Description
0x00000002 ERROR_FILE_NOT_FOUND	The system cannot find the specified file.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000426 ERROR_SERVICE_NOT_ACTIVE	The service is not running.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server MAY return a list of the clients connected to the protocol server if the return code indicates success.<15>

3.1.5.16 R_IISDisconnectUser (Opnum 15)

The **R_IISDisconnectUser** method is called by the client. In response, the server disconnects the specified user from the specified Internet protocol server.

```
DWORD R_IISDisconnectUser(
  [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
  [in] DWORD dwServiceId,
  [in] DWORD dwInstance,
  [in] DWORD dwIdUser
);
```

pszServer: The custom binding handle for the target system, as specified in section 2.1.1. The value MUST NOT be used by the server implementation.

dwServiceId: The identifier for the specified Internet protocol server, as specified in section 2.2.2.

dwInstance: The ID of the Internet protocol server instance whose user is being disconnected.

dwIdUser: The identifier of the user to disconnect, as found in the **idUser** field of the IIS_USER_INFO_1 structure returned by the R_IISEnumerateUsers method. A value of 0 for this parameter indicates that the server implementation MUST attempt to disconnect all users from this Internet protocol server instance.

Return Values: The method returns 0 (ERROR_SUCCESS) to indicate success; otherwise, it returns a nonzero error code, as specified in [MS-ERREF] section 2.2 or[MS-ERREF] section 2.3.1. The most common error codes are listed in the following table.

Return value/code	Description
0x00000002 ERROR_FILE_NOT_FOUND	The system cannot find the specified file.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000426 ERROR_SERVICE_NOT_ACTIVE	The service is not running.
0x000008AD NERR_UserNotFound	The user name could not be found.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE].

In response to this request from the client, the server MAY disconnect a specific user if the *dwIdUser* value specifies that user and SHOULD attempt to disconnect all users if *dwIdUser* is 0.<16>

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

4 Protocol Examples

This section describes a sequence of operations to illustrate the function of the Internet Information Services (IIS) Inetinfo Remote Protocol.

1. The client receives a request from an application to retrieve the virtual root directory of the FTP server on the server host.

2. The client sends an R_InetInfoGetAdminInformation message to the server by first invoking the **R_InetInfoGetAdminInformation** RPC client Stub method with the following parameters:

   - *pszServer* = "server host name"
   - *dwServerMask* = 0x00000001 (INET_FTP_SVC_ID)
   - *ppConfig* = address of LPINET_INFO_CONFIG_INFO

3. The client establishes a connection to the remote server by building an explicit server binding handle as specified in section 2.2.1.

4. When the server receives this request from the client, it allocates and initializes the INET_INFO_CONFIG_INFO structure and populates it with data from the FTP server. The server then returns 0 (ERROR_SUCCESS) and the pointer to the **INET_INFO_CONFIG_INFO** structure in the *ppConfig* parameter of the response.

5. The client retrieves the virtual root data from the returned **INET_INFO_CONFIG_INFO** structure, returns it to the application, and frees the data allocated by the RPC client Stub.

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

Security parameter	Section
RPC_C_AUTHN_WINNT	2.1.1
RPC_C_AUTHN_LEVEL_CONNECT	2.1.2
RPC_C_AUTHN_LEVEL_PKT_PRIVACY	2.1.2

6 Appendix A: Full IDL

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

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

[
    uuid(82ad4280-036b-11cf-972c-00aa006887b0),
    version(2.0),
    ms_union,
    pointer_default(unique)
]
interface inetinfo
{


typedef [handle] [string] LPWSTR INET_INFO_IMPERSONATE_HANDLE;

typedef struct _INET_INFO_CAP_FLAGS {
    DWORD   Flag;
    DWORD   Mask;
} INET_INFO_CAP_FLAGS,
 * LPINET_INFO_CAP_FLAGS;

typedef struct _INET_INFO_CAPABILITIES_STRUCT {
    DWORD   CapVersion;
    DWORD   ProductType;
    DWORD   MajorVersion;
    DWORD   MinorVersion;
    DWORD   BuildNumber;
    DWORD   NumCapFlags;

    [size_is(NumCapFlags)]  LPINET_INFO_CAP_FLAGS   CapFlags;

} INET_INFO_CAPABILITIES_STRUCT,
 * LPINET_INFO_CAPABILITIES_STRUCT;

typedef struct _INET_LOG_CONFIGURATION {
    DWORD   inetLogType;
    DWORD   ilPeriod;
    WCHAR   rgchLogFileDirectory[260];
    DWORD   cbSizeForTruncation;
    WCHAR   rgchDataSource[260];
    WCHAR   rgchTableName[30];
    WCHAR   rgchUserName[257];
    WCHAR   rgchPassword[257];
} INET_LOG_CONFIGURATION,
 * LPINET_LOG_CONFIGURATION;

typedef struct _INET_INFO_IP_SEC_ENTRY {
    DWORD       dwMask;
    DWORD       dwNetwork;

} INET_INFO_IP_SEC_ENTRY,
 *LPINET_INFO_IP_SEC_ENTRY;

typedef struct _INET_INFO_IP_SEC_LIST {
    DWORD               cEntries;
    [size_is( cEntries)] INET_INFO_IP_SEC_ENTRY  aIPSecEntry[];
} INET_INFO_IP_SEC_LIST,
 *LPINET_INFO_IP_SEC_LIST;

typedef struct _INET_INFO_VIRTUAL_ROOT_ENTRY {
    [string] LPWSTR  pszRoot;
    [string] LPWSTR  pszAddress;
```

```
    [string] LPWSTR  pszDirectory;
    DWORD   dwMask;
    [string] LPWSTR  pszAccountName;
    WCHAR   AccountPassword[257];
    DWORD   dwError;
} INET_INFO_VIRTUAL_ROOT_ENTRY,
 *LPINET_INFO_VIRTUAL_ROOT_ENTRY;

typedef struct _INET_INFO_VIRTUAL_ROOT_LIST
{
    DWORD               cEntries;
    [size_is( cEntries)]
        INET_INFO_VIRTUAL_ROOT_ENTRY  aVirtRootEntry[];
} INET_INFO_VIRTUAL_ROOT_LIST,
 *LPINET_INFO_VIRTUAL_ROOT_LIST;

typedef struct _INET_INFO_CONFIG_INFO {
    DWORD                           FieldControl;
    DWORD                           dwConnectionTimeout;
    DWORD                           dwMaxConnections;
    [string] LPWSTR                 lpszAdminName;
    [string] LPWSTR                 lpszAdminEmail;
    [string] LPWSTR                 lpszServerComment;
    LPINET_LOG_CONFIGURATION        lpLogConfig;
    WORD                            LangId;
    LCID                            LocalId;
    BYTE                            ProductId[64];
    BOOL                            fLogAnonymous;
    BOOL                            fLogNonAnonymous;
    [string] LPWSTR                 lpszAnonUserName;
    WCHAR                           szAnonPassword[257];
    DWORD                           dwAuthentication;
    short                           sPort;
    LPINET_INFO_IP_SEC_LIST         DenyIPList;
    LPINET_INFO_IP_SEC_LIST         GrantIPList;
    LPINET_INFO_VIRTUAL_ROOT_LIST   VirtualRoots;
} INET_INFO_CONFIG_INFO,
 * LPINET_INFO_CONFIG_INFO;

typedef struct _INET_INFO_SITE_ENTRY {
    [string] LPWSTR  pszComment;
    DWORD   dwInstance;
} INET_INFO_SITE_ENTRY,
 *LPINET_INFO_SITE_ENTRY;

typedef struct _INET_INFO_SITE_LIST {
    DWORD               cEntries;
    [size_is( cEntries)] INET_INFO_SITE_ENTRY  aSiteEntry[];
} INET_INFO_SITE_LIST,
 *LPINET_INFO_SITE_LIST;

typedef struct _INET_INFO_GLOBAL_CONFIG_INFO {
    DWORD         FieldControl;
    DWORD         BandwidthLevel;
    DWORD         cbMemoryCacheSize;
} INET_INFO_GLOBAL_CONFIG_INFO,
 * LPINET_INFO_GLOBAL_CONFIG_INFO;

typedef struct _INETA_CACHE_STATISTICS {
    DWORD FilesCached;
    DWORD TotalFilesCached;
    DWORD FileHits;
    DWORD FileMisses;
    DWORD FileFlushes;
    DWORDLONG CurrentFileCacheSize;
    DWORDLONG MaximumFileCacheSize;
    DWORD FlushedEntries;
    DWORD TotalFlushed;
    DWORD URICached;
    DWORD TotalURICached;
```

```
    DWORD URIHits;
    DWORD URIMisses;
    DWORD URIFlushes;
    DWORD TotalURIFlushed;
    DWORD BlobCached;
    DWORD TotalBlobCached;
    DWORD BlobHits;
    DWORD BlobMisses;
    DWORD BlobFlushes;
    DWORD TotalBlobFlushed;
} INETA_CACHE_STATISTICS,
 *LPINETA_CACHE_STATISTICS;

typedef struct _INETA_ATQ_STATISTICS {
    DWORD         TotalBlockedRequests;
    DWORD         TotalRejectedRequests;
    DWORD         TotalAllowedRequests;
    DWORD         CurrentBlockedRequests;
    DWORD         MeasuredBandwidth;
} INETA_ATQ_STATISTICS,
 *LPINETA_ATQ_STATISTICS;

typedef struct _INET_INFO_STATISTICS_0 {
    INETA_CACHE_STATISTICS  CacheCtrs;
    INETA_ATQ_STATISTICS    AtqCtrs;
    DWORD   nAuxCounters;
    DWORD   rgCounters[20];
} INET_INFO_STATISTICS_0,
 * LPINET_INFO_STATISTICS_0;

typedef [switch_type(unsigned long)]
union _INET_INFO_STATISTICS_INFO {
    [case(0)]
        LPINET_INFO_STATISTICS_0 InetStats0;
    [default]
        ;
} INET_INFO_STATISTICS_INFO,
 *LPINET_INFO_STATISTICS_INFO;

typedef struct _W3_STATISTICS_1 {
    LARGE_INTEGER TotalBytesSent;
    LARGE_INTEGER TotalBytesReceived;
    DWORD         TotalFilesSent;
    DWORD         TotalFilesReceived;
    DWORD         CurrentAnonymousUsers;
    DWORD         CurrentNonAnonymousUsers;
    DWORD         TotalAnonymousUsers;
    DWORD         TotalNonAnonymousUsers;
    DWORD         MaxAnonymousUsers;
    DWORD         MaxNonAnonymousUsers;
    DWORD         CurrentConnections;
    DWORD         MaxConnections;
    DWORD         ConnectionAttempts;
    DWORD         LogonAttempts;
    DWORD         TotalOptions;
    DWORD         TotalGets;
    DWORD         TotalPosts;
    DWORD         TotalHeads;
    DWORD         TotalPuts;
    DWORD         TotalDeletes;
    DWORD         TotalTraces;
    DWORD         TotalMove;
    DWORD         TotalCopy;
    DWORD         TotalMkcol;
    DWORD         TotalPropfind;
    DWORD         TotalProppatch;
    DWORD         TotalSearch;
    DWORD         TotalLock;
    DWORD         TotalUnlock;
    DWORD         TotalOthers;
```

```
    DWORD         TotalCGIRequests;
    DWORD         TotalBGIRequests;
    DWORD         TotalNotFoundErrors;
    DWORD         TotalLockedErrors;
    DWORD         CurrentCalAuth;
    DWORD         MaxCalAuth;
    DWORD         TotalFailedCalAuth;
    DWORD         CurrentCalSsl;
    DWORD         MaxCalSsl;
    DWORD         TotalFailedCalSsl;
    DWORD         CurrentCGIRequests;
    DWORD         CurrentBGIRequests;
    DWORD         MaxCGIRequests;
    DWORD         MaxBGIRequests;
    DWORD         CurrentBlockedRequests;
    DWORD         TotalBlockedRequests;
    DWORD         TotalAllowedRequests;
    DWORD         TotalRejectedRequests;
    DWORD         MeasuredBw;
    DWORD         ServiceUptime;
    DWORD         TimeOfLastClear;
    DWORD         nAuxCounters;
    DWORD         rgCounters[20];
} W3_STATISTICS_1,
 * LPW3_STATISTICS_1;

typedef [switch_type(unsigned long)]
union _W3_STATISTICS_UNION {
    [case(0)]
       LPW3_STATISTICS_1 StatInfo1;
    [default]
       ;
} W3_STATISTICS_STRUCT,
 *LPW3_STATISTICS_STRUCT;

typedef struct _FTP_STATISTICS_0 {
    LARGE_INTEGER TotalBytesSent;
    LARGE_INTEGER TotalBytesReceived;
    DWORD         TotalFilesSent;
    DWORD         TotalFilesReceived;
    DWORD         CurrentAnonymousUsers;
    DWORD         CurrentNonAnonymousUsers;
    DWORD         TotalAnonymousUsers;
    DWORD         TotalNonAnonymousUsers;
    DWORD         MaxAnonymousUsers;
    DWORD         MaxNonAnonymousUsers;
    DWORD         CurrentConnections;
    DWORD         MaxConnections;
    DWORD         ConnectionAttempts;
    DWORD         LogonAttempts;
    DWORD         ServiceUptime;
    DWORD         TotalAllowedRequests;
    DWORD         TotalRejectedRequests;
    DWORD         TotalBlockedRequests;
    DWORD         CurrentBlockedRequests;
    DWORD         MeasuredBandwidth;
    DWORD         TimeOfLastClear;
} FTP_STATISTICS_0,
 * LPFTP_STATISTICS_0;

typedef [switch_type(unsigned long)]
union _FTP_STATISTICS_UNION {
    [case(0)]
       LPFTP_STATISTICS_0 StatInfo0;
    [default]
       ;
} FTP_STATISTICS_STRUCT,
 *LPFTP_STATISTICS_STRUCT;

typedef struct _IIS_USER_INFO_1 {
```

```
    DWORD           idUser;
    [string] LPWSTR pszUser;
    BOOL            fAnonymous;
    DWORD           inetHost;
    DWORD           tConnect;
} IIS_USER_INFO_1,
 * LPIIS_USER_INFO_1;

typedef struct _IIS_USER_INFO_1_CONTAINER {
    DWORD EntriesRead;
    [size_is(EntriesRead)] LPIIS_USER_INFO_1 Buffer;
} IIS_USER_INFO_1_CONTAINER,
 *LPIIS_USER_INFO_1_CONTAINER;

typedef struct _IIS_USER_ENUM_STRUCT {
    DWORD Level;
    [switch_is(Level)] union _USER_ENUM_UNION
    {
        [case(1)]
            LPIIS_USER_INFO_1_CONTAINER Level1;
        [default]
            ;
    } ConfigInfo;
} IIS_USER_ENUM_STRUCT,
 *LPIIS_USER_ENUM_STRUCT;

DWORD
R_InetInfoGetVersion(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
    [in] DWORD dwReserved,
    [out] DWORD *pdwVersion
    );

DWORD
R_InetInfoGetAdminInformation(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
    [in] DWORD dwServerMask,
    [out] LPINET_INFO_CONFIG_INFO *ppConfig
    );

DWORD
R_InetInfoGetSites(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
    [in] DWORD dwServerMask,
    [out] LPINET_INFO_SITE_LIST * ppSites
    );

DWORD
R_InetInfoSetAdminInformation(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
    [in] DWORD dwServerMask,
    [in, ref] INET_INFO_CONFIG_INFO *pConfig
    );

DWORD
R_InetInfoGetGlobalAdminInformation(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE     pszServer,
    [in] DWORD dwServerMask,
    [out] LPINET_INFO_GLOBAL_CONFIG_INFO *ppConfig
    );

DWORD
R_InetInfoSetGlobalAdminInformation(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE   pszServer,
    [in] DWORD dwServerMask,
    [in, ref] INET_INFO_GLOBAL_CONFIG_INFO * pConfig
    );

DWORD
R_InetInfoQueryStatistics(
```

```
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE  pszServer,
    [in] DWORD Level,
    [in] DWORD dwServerMask,
    [out, switch_is(Level)] LPINET_INFO_STATISTICS_INFO StatsInfo
    );

DWORD
R_InetInfoClearStatistics(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE  pszServer,
    [in] DWORD dwServerMask
    );

DWORD
R_InetInfoFlushMemoryCache(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE  pszServer,
    [in] DWORD dwServerMask
    );

DWORD
R_InetInfoGetServerCapabilities(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
    [in] DWORD dwReserved,
    [out] LPINET_INFO_CAPABILITIES_STRUCT *ppCap
    );

DWORD
R_W3QueryStatistics2(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
    [in] DWORD dwLevel,
    [in] DWORD dwInstance,
    [in] DWORD dwReserved,
    [out,switch_is(dwLevel)] LPW3_STATISTICS_STRUCT   InfoStruct
    );

DWORD
R_W3ClearStatistics2(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
    [in] DWORD dwInstance
    );

DWORD
R_FtpQueryStatistics2(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
    [in] DWORD dwLevel,
    [in] DWORD dwInstance,
    [in] DWORD dwReserved,
    [out,switch_is(dwLevel)] LPFTP_STATISTICS_STRUCT   InfoStruct
    );

DWORD
R_FtpClearStatistics2(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
    [in] DWORD dwInstance
    );

DWORD
R_IISEnumerateUsers(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
    [in] DWORD dwServiceId,
    [in] DWORD dwInstance,
    [in,out] LPIIS_USER_ENUM_STRUCT InfoStruct
    );

DWORD
R_IISDisconnectUser(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
    [in] DWORD dwServiceId,
    [in] DWORD dwInstance,
    [in] DWORD dwIdUser
    );
```

```
    DWORD
    Opnum16NotUsedOnWire();

    DWORD
    Opnum17NotUsedOnWire();
    }
```

7 (Updated Section) Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

- Windows NT 4.0 operating system
- Windows 2000 Professional operating system
- Windows 2000 Server operating system
- Windows XP operating system
- Windows Server 2003 operating system
- Windows Vista operating system
- Windows Server 2008 operating system
- Windows 7 operating system
- Windows Server 2008 R2 operating system
- Windows 8 operating system
- Windows Server 2012 operating system
- Windows 8.1 operating system
- Windows Server 2012 R2 operating system
- Windows 10 operating system
- Windows Server 2016 operating system
- Windows Server operating system
- Windows Server 2019 operating system

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates 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.

<1> Section 1.7: The Internet Information Services (IIS) Inetinfo Remote Protocol is implemented in Windows by Internet Information Services (IIS). The RPC interface and data types have been modified between versions of IIS (without changing the version number) in ways that make complete interoperability between versions difficult if not impossible.

The version described as the standard in this document is IIS version 5.0 that shipped with Windows 2000 Professional, and Windows 2000 Server.

<2> Section 2.1.1: When the Windows implementation specifies "ncacn_np" as the RPC protocol, a security descriptor is set on the endpoint (2) by using the RPC API RpcServerUseProtseqEpW(). The

security descriptor grants access to members of the administrators group. The RPC runtime will then validate the client's identity against this security descriptor.

<3> Section 2.2.3: Some members of the INET_INFO_CONFIG_INFO structure are not persisted such as **LangId** and **LocalId**. These vary by version and by protocol.

<4> Section 2.2.3: The Windows implementation will return an empty string when retrieving **szAnonPassword**, even if the configuration value is set.

<5> Section 2.2.8: The Windows implementation will return an empty string when retrieving **AccountPassword**, even if the configuration value is set.

<6> Section 2.2.15: The INETA_CACHE_STATISTICS structure has been modified twice in different versions of IIS.

IIS version 3.0 of the structure is defined as follows:

```
typedef struct _INETA_CACHE_STATISTICS
  {
   DWORD         CacheBytesTotal;
   DWORD         CacheBytesInUse;
   DWORD         CurrentOpenFileHandles;
   DWORD         CurrentDirLists;
   DWORD         CurrentObjects;
   DWORD         FlushesFromDirChanges;
   DWORD         CacheHits;
   DWORD         CacheMisses;
 } INETA_CACHE_STATISTICS * LPINETA_CACHE_STATISTICS;
```

CacheBytesTotal: The total size of the cache in bytes.

CacheBytesInUse: The number of bytes in the cache currently in use.

CurrentOpenFileHandles: The number of handles to currently open files stored in the cache.

CurrentDirLists: The number of current directory lists stored in the cache.

CurrentObjects: The number of current objects stored in the cache.

FlushesFromDirChanges: The number of flushes that have taken place as a result of directory changes.

CacheHits: The number of hits to the cache.

CacheMisses: The number of misses to the cache.

IIS version 6.0 of the structure is defined as follows:

```
typedef struct _INETA_CACHE_STATISTICS
 {
    DWORD FilesCached;
    DWORD TotalFilesCached;
    DWORD FileHits;
    DWORD FileMisses;
    DWORD FileFlushes;
    DWORDLONG CurrentFileCacheSize;
    DWORDLONG MaximumFileCacheSize;
    DWORD FlushedEntries;
```

```
    DWORD TotalFlushed;
    DWORD URICached;
    DWORD TotalURICached;
    DWORD URIHits;
    DWORD URIMisses;
    DWORD URIFlushes;
    DWORD TotalURIFlushed;
    DWORD BlobCached;
    DWORD TotalBlobCached;
    DWORD BlobHits;
    DWORD BlobMisses;
    DWORD BlobFlushes;
    DWORD TotalBlobFlushed;
} INETA_CACHE_STATISTICS, *LPINETA_CACHE_STATISTICS;
```

CurrentFileCacheSize and **MaximumFileCacheSize** are the only fields changed from the IIS version 5.0. Their data types have been changed from DWORD to DWORDLONG.

<7> Section 2.2.25: The Windows implementation can return error codes that have additional semantic meaning because it relates to the protocol beyond that which is specified in [MS-ERREF]. The following table summarizes these additional semantics.

Return value/code	Interpretation/condition
0x00000426 ERROR_SERVICE_NOT_ACTIVE	Used to indicate that the specified Internet protocol service is not recognized by the implementation.
0x00000002 ERROR_FILE_NOT_FOUND	Used by methods that operate on an Internet protocol server instance to indicate that the instance ID passed by the client is not recognized.

<8> Section 3.1.5: Opnums reserved for local use apply to Windows as follows.

Opnum	Description
16, 17	Only used locally by Windows, never remotely. These methods were added to the inetinfo interface in Windows 2000 Server and deprecated in Windows Server 2003.

<9> Section 3.1.5.1: The Windows implementation of R_InetInfoGetVersion does not return Major Version = 5 and Minor Version = 1 for implementations of IIS released on Windows NT 4.0, Windows 2000 Professional, and Windows 2000 Server.

<10> Section 3.1.5.3: In IIS version 3.0, the R_InetInfoGetSites method is not implemented. Instead, the following method is defined at opnum 2.

```
R_InetInfoDummy(
    [in, string, unique] INET_INFO_IMPERSONATE_HANDLE pszServer,
    [in] DWORD dwServerMask,
    [out] LPINET_INFO_CONFIG_INFO *ppConfig
    );
```

This method returns 0 and does nothing.

<11> Section 3.1.5.7: The Windows implementation of R_InetInfoQueryStatistics returns valid statistical data only when the *dwServerMask* parameter is 0.

<12> Section 3.1.5.8: The Windows implementation of R_InetInfoClearStatistics does nothing and returns ERROR_NOT_SUPPORTED. If *dwServerMask* is greater than 0x00000004, the return code is ERROR_INVALID_PARAMETER.

<13> Section 3.1.5.12: The Windows implementation of R_W3ClearStatistics2 does not reset all statistics values in the W3_STATISTICS_1 structure. Values related to client network connections and network bandwidth are not reset.

<14> Section 3.1.5.14: The Windows implementation of R_FtpClearStatistics2 does not reset all statistics values in the FTP_STATISTICS_0 structure. Values related to client network connections and network bandwidth are not reset.

<15> Section 3.1.5.15: The Windows implementation of R_IISEnumerateUsers returns ERROR_SUCCESS (0x00000000) for the HTTP server when the method is not implemented.

<16> Section 3.1.5.16: The Windows implementation of R_IISDisconnectUser for the FTP server will attempt to disconnect specific users in addition to all users, given appropriate values of *dwIdUser*. The HTTP server will only attempt to disconnect all users if *dwIdUser* is 0 and will ignore other values of this parameter and return ERROR_SUCCESS.

8 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Description	Revision class
7 Appendix B: Product Behavior	Added Windows Server 2019 to the list of applicable products.	Major

9 Index

L

M

N

O

P

R

S

T

V

W