[MS-RAIW]:
Remote Administrative Interface:
WINS Specification

Revision Summary

Date	Revision History	Revision Class	Comments
03/12/2010	1.0	Major	First Release.
04/23/2010	1.0.1	Editorial	Revised and edited the technical content.
06/04/2010	1.1	Minor	Updated the technical content.
07/16/2010	1.1	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	1.1	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	1.2	Minor	Clarified the meaning of the technical content.
11/19/2010	2.0	Major	Significantly changed the technical content.
01/07/2011	2.1	Minor	Clarified the meaning of the technical content.
02/11/2011	2.1	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This is a specification of the Remote Administrative Interface: WINS protocol. This protocol defines **remote procedure call (RPC)** interfaces that provide methods for remotely accessing and administering a server for the **Windows Internet Name Service (WINS)**. This protocol is a client/server protocol that is based on RPC and is used in the configuration, management, and monitoring of a **WINS server**.

An application implementing this protocol can remotely perform service monitoring of a WINS server as well as creating, updating, querying, or deleting database records, performing database **scavenging**, and **replicating** the database records with other WINS servers.

1.1 Glossary

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

authentication level
Authentication Service (AS)
broadcast
client (1)
domain
domain controller (DC)
domain master browser server
domain name
Domain Name System (DNS)
dynamic endpoint
endpoint
globally unique identifier (GUID)
Interface Definition Language (IDL)
Internet Protocol version 4 (IPv4)
Internet Protocol version 6 (IPv6)
IPv4 address in string format
little-endian
Microsoft Interface Definition Language (MIDL)
multicast
named pipe
NetBIOS
NetBIOS name
NetBIOS Name Server (NBNS)
NetBIOS suffix
Network Data Representation (NDR)
opnum
remote procedure call (RPC)
replication
RPC protocol sequence
RPC transfer syntax
RPC transport
security provider
Transmission Control Protocol (TCP)
unicast
Unicode
Unicode string
UTF-16LE (Unicode Transformation Format, 16-bits, little-endian)

universally unique identifier (UUID)
well-known endpoint

The following terms are specific to this document:

ACTIVE: The state of a **name record**, in which it has been registered but not released.

active record: A **name record** that has been registered but not released.

address version map: See **owner version map**.

attributes of a name record: The name, type of record, IP Address, version number, node type, owner id, state, static type, and time stamp of a **name record**.

browser name: A **NetBIOS name** whose 16th character is set to 0x1B. This name is used to identify the **domain master browser server** for a **domain**.

dynamic record: A **name record** that was created through **NetBT** name registration by a **client**.

extinction interval: The interval at which released names are changed to the **tombstone state**.

HostName: The name of a host on a network. Users specify computers on a network by their host names.

multihomed: Multiple network interfaces to multiple separate physical networks and thus multiple **IPv4** addresses.

multihomed machine name: The **NetBIOS name** of a machine that is **multihomed**.

name record: The **NetBIOS name**-to-**IPv4** address mapping.

name server: The server that resolves names for hosts by providing **NetBIOS name**-to-**IPv4** address mappings.

NBNS partner: An **NBNS replication partner**.

NBNS pull partner: A **NetBIOS name server** that requests new **NBNS name records** (**replicas**) from its **partner**.

NBNS push partner: An **NBNS server** that pushes or notifies **NBNS servers** that are configured as **pull partners** of the need to **replicate** their **name records**.

NBNS replication partner: An **NBNS server** that is configured or discovered as a partner to exchange the **NBNS** database.

NetBIOS scope: The population of computers across which a registered **NetBIOS name** is known. Each NetBIOS scope has a scope identifier, which is a character string that meets the requirements of the **Domain Name System (DNS)** for **domain names**.

NetBT: **NetBIOS** over TCP/IP.

NetBT b-node: A **NetBT node type** that is configured to use **broadcast NetBIOS name** queries for name registration and resolution.

NetBT h-node: A combination of **NetBT b-node** and **p-node** functionality. An h-node uses point-to-point communication first. If the **NBNS** cannot be located, h-node switches to

broadcast. The h-node continues to poll for the name server and returns to point-to-point communication when one becomes available.

NetBT m-node: A **NetBT node type** that uses a mix of **b-node** and **p-node** communications to register and resolve **NetBIOS names**. An m-node uses **broadcast** resolution first; then, if necessary, it uses a server query.

NetBT name resolution: The process of resolving a **NetBIOS name** to an **IPv4** address.

NetBT node type: The transport mechanism used to resolve **NetBIOS names** that are **broadcast**, **multicast**, or **unicast**.

NetBT p-node: A **NetBT node type** that does not use **broadcasts** for name registration or **name_resolution**. Instead, all systems register themselves with an **NBNS** upon startup. The **NBNS** is responsible for mapping computer names to **IPv4** addresses and making sure that no duplicate names are registered on the network.

normal group: A group of hosts that does not have an associated address. It is assumed to be valid on any subnet.

owner NBNS server: An **NBNS server** that handles the name registration of a **client** and so owns the mapping for that **client**. An owner NBNS server is also referred to by the term **owner WINS server** in this document.

owner version map: A table in which each entry has two fields, owner and version number. The owner field contains a **WINS server** address; the version number field contains the highest version number of all the records owned by the **owner WINS server** that are stored at the local **WINS server**.

owner WINS server: See **owner NBNS server**.

partner: See **NBNS partner**

point-to-point node: See **NetBT p-node**.

priority class: An attribute of a process that is used to determine the scheduling priority of threads of that process. The priority of a thread is determined by a combination of the **priority class** of its process and the priority level of the thread within the **priority class**.

pull partner: See **NBNS pull partner**

RELEASED: The state of a **name record**, in which its name has been explicitly released through a name release request, or in which it has failed to be refreshed by name by a **client** within the renewal interval.

released record: A **name record** that has been explicitly released through a name release request; or a **name record** that a **client** has failed to refresh by name within the renewal interval.

replica: **NBNS** database **name records** (name-to-**IPv4** address mapping) **replicated** from other **NBNS servers**.

replication partner: See **NBNS replication partner**

scavenging: The process in which the state of database records is changed if the current time exceeds the record's time stamp value. For example, an **active record** becomes a **released record** when the current time exceeds the time stamp associated with that record.

special group: A group of hosts that have a single name. When a name registration is received for a special group, the actual address rather than the limited **broadcast** address is stored in the group. When a name query is received for such a group, the **IPv4** addresses that have not timed out are returned.

static record: A manually created entry in the database of a **NBNS server**.

target WINS server: The **WINS server** on which the **RPC** method call is being executed.

time to live (TTL): The amount of time that a **NetBIOS name** is stored on an **NBNS server**.

tombstone interval: See **extinction interval**.

tombstone state, tombstoned: The state of a **released record** that is not re-registered or refreshed by a **client** within the **extinction interval**.

Windows Internet Name Service (WINS): The Microsoft implementation of an **NBNS server**.

WINS server: A server that hosts a Microsoft implementation of an **NBNS server**.

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

1.2 References

1.2.1 Normative References

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

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

[ISO-8601] International Organization for Standardization, "Data Elements and Interchange Formats - Information Interchange - Representation of Dates and Times", ISO/IEC 8601:2004, December 2004, http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=40874&ICS1=1&ICS2=140&ICS3=30

Note There is a charge to download the specification.

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

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

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions", July 2006.

[MS-WINSRA] Microsoft Corporation, "Windows Internet Naming Service (WINS) Replication and Autodiscovery Protocol Specification" July 2007.

[RFC1002] Network Working Group, "Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications", STD 19, RFC 1002, March 1987, http://www.ietf.org/rfc/rfc1002.txt

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

1.2.2 Informative References

[LMHOSTS] Microsoft Corporation, "LMHOSTS File Information and Predefined Keywords", February 2007, http://support.microsoft.com/kb/102725

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

[MSDN-Handles] Microsoft Corporation, "Handles", http://msdn.microsoft.com/en-us/library/aa373932(VS.85).aspx

If you have any trouble finding [MSDN-Handles], please check here.

1.3 Overview

The Remote Administrative Interface: WINS protocol is a client/server protocol that is used to remotely configure, manage, and monitor the WINS server. This protocol allows a **client** to view and update the server configuration settings as well as to create, modify, and delete WINS database records. It also allows clients to trigger scavenging and replicating operations and to query the database.

The Remote Administrative Interface: WINS protocol is stateless with no state shared across RPC method calls. Each RPC method call contains one complete request. Output from one method call can be used as an input to another call, but the protocol does not provide methods for locking the WINS server configuration or state data across method calls. For example, a client can pull a range of records from the database and delete some of them using another RPC call. However, the protocol does not guarantee that the specified record has not been modified by another client between the two method calls.

<table>
<tr><th colspan="2">Remote Administrative Interface: WINS</th></tr>
<tr><td colspan="2">Remote Procedure Call (RPC)</td></tr>
<tr><td>TCP</td><td>Named Pipes</td></tr>
</table>

Figure 1: Relationship of Remote Administrative Interface: WINS to RPC

1.4 Relationship to Other Protocols

The Remote Administrative Interface: WINS protocol relies on RPC [MS-RPCE] as a transport. It is used to manage the WINS service on servers that implement the Windows Internet Naming Service (WINS) Replication and Autodiscovery Protocol [MS-WINSRA].

1.5 Prerequisites/Preconditions

The Remote Administrative Interface: WINS protocol is implemented on top of RPC and, as a result, has the prerequisites identified in [MS-RPCE].

The Remote Administrative Interface: WINS protocol assumes that before this protocol is invoked, a client has obtained the name or the IP address of the WINS server that implements this protocol suite.

1.6 Applicability Statement

The Remote Administrative Interface: WINS protocol is applicable when an application needs to remotely configure, manage, or monitor a WINS server.

Because the **NetBIOS** protocol [RFC1002] does not support the mapping between **NetBIOS names** and **IPv6** addresses, the Remote Administrative Interface: WINS protocol applies only to **IPv4** addresses. It does not apply to IPv6 addresses.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Supported Transports**: The Remote Administrative Interface: WINS protocol uses the RPC protocol as a transport and RPC Protocol Sequences as specified in section 2.1.
- **Protocol Versions**: This protocol has only one interface version, but that interface has been extended by adding additional methods at the end. The use of these methods is specified in section 3.1.
- **Security and Authentication Methods**: Authentication and security for the methods specified by this protocol are specified in [MS-RPCE] and in section 2.1.
- **Localization**: This protocol passes text strings in various methods. Localization considerations for such strings are specified in sections 2.2 and 3.1.4.
- **Capability Negotiation**: The Remote Administrative Interface: WINS protocol does not support interface version negotiation. Instead, this protocol uses the interface version number specified in the **interface definition language (IDL)** for versioning and capability negotiation.

1.8 Vendor-Extensible Fields

The Remote Administrative Interface: WINS protocol uses Win32 error codes as defined in [MS-ERREF] (section 2.2). Vendors SHOULD reuse those values with their indicated meanings. Choosing any other value runs the risk of a collision in the future.

1.9 Standards Assignments

Remote Administrative Interface: WINS protocol uses the following private assignments.

Parameter	Value	Reference
RPC interface **universally unique identifier (UUID)**	45f52c28-7f9f-101a-b52b-08002b2efabe	[C706] section A.2.5 winsif interface (section 3.1)
RPC interface UUID	811109bf-a4e1-11d1-ab54-00a0c91e9b45	[C706] section A.2.5 winsi2 interface (section 3.2)
Named pipe	"\pipe\WinsPipe"	Transport (section 2.1)

2 Messages

2.1 Transport

For the Remote Administrative Interface: WINS protocol, the WINS server MUST support the following RPC transports:

- RPC over **TCP**, with port selection performed dynamically by RPC.
- RPC over named pipes, with the **endpoint** name "\pipe\WinsPipe".

2.1.1 Server Security Settings

The Remote Administrative Interface: WINS protocol uses **security support provider (SSP)** security provided by RPC as specified in [MS-RPCE]. The WINS RPC server uses the principal name "Wins" and the authentication service RPC_C_AUTHN_WINNT.

The WINS server MUST allow only authenticated access to RPC clients. The WINS server MUST NOT allow anonymous or unauthenticated RPC clients to connect. The WINS server MUST perform authorization checks to ensure that the client is authorized to perform a specific RPC operation.

The following mechanisms are enforced for client authorization:

- The WINSRA client SHOULD be a member of the WINS Users or WINS Administrator security group in order to retrieve information from the WINS server. This level of authorization is termed "query-level access".<1>
- The WINSRA client MUST be a member of the WINS Administrator security group before it can retrieve or modify information on the WINS server. This level of authorization is termed "control-level access".

Control-level access also includes query-level access. Therefore, clients with control access can also call methods that require only query-level access. The WINS server MUST limit access to only those clients that negotiate an authentication level equal to or higher than RPC_C_AUTHN_LEVEL_CONNECT.

2.1.2 Client Security Settings

The RPC client SHOULD use security support provider (SSP) security provided by RPC as specified in [MS-RPCE]. The clients SHOULD use the server principal name "Wins", authentication service RPC_C_AUTHN_WINNT, and authentication level RPC_C_AUTHN_LEVEL_CONNECT while creating the binding handle.

2.2 Common Data Types

In addition to the RPC base types and definitions specified in [C706] and [MS-RPCE], additional data types are defined in this section. The following statements apply to those data types unless indicated otherwise:

- All multiple byte numeric values in messages use **little-endian** byte order.
- All character strings are encoded in **Unicode UTF-16LE**.

2.2.1 Datatypes, Enumerations, and Constants

2.2.1.1 WINSIF_HANDLE

The **WINSIF_HANDLE** data type is defined as a pointer to the **WINSINTF_BIND_DATA_T** structure. It is used by the RPC methods **R_WinsGetBrowserNames** and **R_WinsStatusWHdl**.

This type is declared as follows:

```
typedef [handle] PWINSINTF_BIND_DATA_T WINSIF_HANDLE;
```

2.2.1.2 WINSINTF_VERS_NO_T

The **WINSINTF_VERS_NO_T** data type indicates the version number of a WINS database record. It is used by several RPC methods like **R_WinsGetDbRecs** and **R_WinsDelDbRecs**.

This type is declared as follows:

```
typedef LARGE_INTEGER WINSINTF_VERS_NO_T;
```

2.2.1.3 WINSINTF_MAX_NO_RPL_PNRS

The **WINSINTF_MAX_NO_RPL_PNRS** constant defines the maximum number of **pull replication partners**. It is used by the structure **WINSINTF_RESULTS_T (section 2.2.2.7)**.

Constant/value	Description
WINSINTF_MAX_NO_RPL_PNRS 25	The maximum number of pull replication partners.

2.2.1.4 WINSINTF_ACT_E

The **WINSINTF_ACT_E** enumeration indicates an action type requested by the RPC method **R_WinsRecordAction** for a record contained in the **WINSINTF_RECORD_ACTION_T** structure.

```
typedef  enum _WINSINTF_ACT_E
{
  WINSINTF_E_INSERT = 0,
  WINSINTF_E_DELETE,
  WINSINTF_E_RELEASE,
  WINSINTF_E_MODIFY,
  WINSINTF_E_QUERY
} WINSINTF_ACT_E,
 *PWINSINTF_ACT_E;
```

WINSINTF_E_INSERT: Insert a record into the WINS database.

WINSINTF_E_DELETE: Delete a matching record from the WINS database.

WINSINTF_E_RELEASE: Release a matching record from the WINS database.

WINSINTF_E_MODIFY: Modify the attributes of the matching record.

WINSINTF_E_QUERY: Query the database for a given name.

2.2.1.5 WINSINTF_CMD_E

The **WINSINTF_CMD_E** enumeration is used by the RPC methods to retrieve the configuration of a particular WINS server. This enumeration is used in conjunction with the **WINSINTF_RESULTS_T** and **WINSINTF_RESULTS_NEW_T** structures.

```
typedef  enum _WINSINTF_CMD_E
{
  WINSINTF_E_ADDVERSMAP = 0,
  WINSINTF_E_CONFIG,
  WINSINTF_E_STAT,
  WINSINTF_E_CONFIG_ALL_MAPS
} WINSINTF_CMD_E,
 *PWINSINTF_CMD_E;
```

WINSINTF_E_ADDVERSMAP: Gets an entry from the **owner version map** of the **target WINS server**.

WINSINTF_E_CONFIG: Get the configuration details of the target WINS server.

WINSINTF_E_STAT: Get statistics for the target WINS server.

WINSINTF_E_CONFIG_ALL_MAPS: Get all owner version map entries from the target WINS server.

2.2.1.6 WINSINTF_TRIG_TYPE_E

The **WINSINTF_TRIG_TYPE_E** enumeration defines the type of replication to be done. It is used by the RPC method **R_WinsTrigger**.

```
typedef  enum _WINSINTF_TRIG_TYPE_E
{
  WINSINTF_E_PULL = 0,
  WINSINTF_E_PUSH,
  WINSINTF_E_PUSH_PROP
} WINSINTF_TRIG_TYPE_E,
 *PWINSINTF_TRIG_TYPE_E;
```

WINSINTF_E_PULL: The target WINS server performs pull replication with the specified WINS server.

WINSINTF_E_PUSH: The target WINS server performs push replication with the specified WINS server.

WINSINTF_E_PUSH_PROP: The target WINS server performs propagating push replication with the specified WINS server.

2.2.1.7 WINSINTF_PRIORITY_CLASS_E

The **WINSINTF_PRIORITY_CLASS_E** enumeration defines the **priority class** of a WINS process. It is used by the RPC method **R_WinsSetPriorityClass**.

```
typedef  enum _WINSINTF_PRIORITY_CLASS_E
{
  WINSINTF_E_NORMAL = 0,
  WINSINTF_E_HIGH
} WINSINTF_PRIORITY_CLASS_E,
 *PWINSINTF_PRIORITY_CLASS_E;
```

WINSINTF_E_NORMAL: WINS process is assigned normal priority class.

WINSINTF_E_HIGH: WINS process is assigned high priority class.

2.2.1.8 WINSINTF_SCV_OPC_E

The **WINSINTF_SCV_OPC_E** enumeration specifies the type of scavenging to be done on the target WINS server. This enumeration is used in the structure **WINSINTF_SCV_REQ_T**.

```
typedef  enum _WINSINTF_SCV_OPC_E
{
  WINSINTF_E_SCV_GENERAL = 0,
  WINSINTF_E_SCV_VERIFY
} WINSINTF_SCV_OPC_E,
 *PWINSINTF_SCV_OPC_E;
```

WINSINTF_E_SCV_GENERAL: Requests normal scavenging operation.

WINSINTF_E_SCV_VERIFY: Verifies only the replicated **active records** with their **owner NBNS servers** for their validity.

2.2.2 Structures

2.2.2.1 WINSINTF_ADD_T

The **WINSINTF_ADD_T** structure defines the IP address information of a WINS server. It is used by several data structures including **WINSINTF_RECORD_ACTION_T** and **WINSINTF_ADD_VERS_MAP_T** and by RPC methods like **R_WinsTrigger** and **R_WinsGetDbRecs**.

```
typedef struct _WINSINTF_ADD_T {
  BYTE Type;
  DWORD Len;
  DWORD IPAdd;
} WINSINTF_ADD_T,
 *PWINSINTF_ADD_T;
```

Type: Specifies the address type. This field MUST be set to zero.

Len: Indicates the length, in bytes, of the IP address that is stored in **IPAdd**.

IPAdd: Stores an IP address in little-endian format. For example, the IP address 172.22.32.42 is stored as 0xAC16202A.

2.2.2.2 WINSINTF_BIND_DATA_T

The **WINSINTF_BIND_DATA_T** structure defines the binding information of the WINS server to which the client connects.

```
typedef struct _WINSINTF_BIND_DATA_T {
  DWORD fTcpIp;
  [string] LPSTR pServerAdd;
  [string] LPSTR pPipeName;
} WINSINTF_BIND_DATA_T,
 *PWINSINTF_BIND_DATA_T;
```

fTcpIp: The transport mechanism to be used. If this value is 0x00000001, then TCP/IP is selected; otherwise, the named pipe is selected.

pServerAdd: A null-terminated string that specifies the server IP address.

pPipeName: A null-terminated string that specifies the pipe name. This value MUST be NULL when *fTcpIP* is 0x00000001.

2.2.2.3 WINSINTF_RECORD_ACTION_T

The **WINSINTF_RECORD_ACTION_T** structure defines a WINS database record and the action to be performed on it. The structure **WINSINTF_RECS_T (section 2.2.2.8)** and the RPC method **R_WinsRecordAction (section 3.1.4.1)** both use this structure.

```
typedef struct _WINSINTF_RECORD_ACTION_T {
  WINSINTF_ACT_E Cmd_e;
  [size_is(NameLen + 1)] LPSTR pName;
  DWORD NameLen;
  DWORD TypOfRec_e;
  DWORD NoOfAdds;
  [unique, size_is(NoOfAdds)] PWINSINTF_ADD_T pAdd;
  WINSINTF_ADD_T Add;
  LARGE_INTEGER VersNo;
  BYTE NodeTyp;
  DWORD OwnerId;
  DWORD State_e;
  DWORD fStatic;
  DWORD_PTR TimeStamp;
} WINSINTF_RECORD_ACTION_T,
 *PWINSINTF_RECORD_ACTION_T;
```

Cmd_e: A **WINSINTF_ACT_E** enumeration (section 2.2.1.4) value that specifies the action to be performed on the specified record.

pName: A pointer to a null-terminated string that contains the NetBIOS name and optionally the **NetBIOS scope** name of the record. The NetBIOS scope name, if present, is appended to the NetBIOS name with a dot character ".".

If the NetBIOS name contains fewer than 16 characters, space characters MUST be used to pad the name string up to the **NetBIOS suffix**, which occupies the 16th character position.

NameLen: The length of the string that *pName* points to. It has the following possible values:

Value	Meaning
16	The **pName** value points to a string that contains only the NetBIOS name of the record. The **NameLen** value does not include the terminating null character.
18 < *value*	The **pName** value points to a string that contains the NetBIOS name, a dot character ".", and the null-terminated NetBIOS scope name of the record. The **NameLen** value includes the terminating null character. If the **NameLen** value is greater than 255, the **pName** string SHOULD be truncated to 254 characters plus a terminating null character.

TypOfRec_e: The record type. Only the two least-significant bits of the member value are considered valid. All other bits are masked with zero. The following values are allowed.

Value	Meaning
0	Unique name
1	Normal group name
2	Special group name
3	**Multihomed** machine name

NoOfAdds: The number of IP addresses that are mapped to the NetBIOS name given in *pName*. It SHOULD have the value zero for unique names and **normal groups**, and it SHOULD have a value greater than 0x00000001 for other types of records.

pAdd: A pointer to an array of IP addresses that are mapped to the name given in *pName*. It MUST be used only for multihomed and **special group** types of records.

Add: The IP address mapped to the name given in *pName*. This member MUST be used only for unique and normal group types of records.

VersNo: The version number of the record.

NodeTyp: The **NetBT node type**. Only the two least-significant bits of the member value are considered valid. All other bits are masked with zero. This member MUST have one of the following values:

Value	Meaning
0	**B-node**
1	**P-node**
2	**M-node**
3	**H-node**

OwnerId: The owner IP address of the record, in little-endian byte order.

State_e: The state of the record. Only the two least-significant bits of the member value are considered valid. All other bits are masked with zero. This member MUST have one of the following values:

Value	Meaning
0	Active record
1	**Released record**
2	**Tombstoned** record
3	Deleted record

fStatic: A value that indicates whether the record is **static** or **dynamic**. A value of 0 indicates a dynamic record, and 1 indicates a static record. Only the least-significant bit is considered valid. All other bits are masked with zero.

TimeStamp: The time stamp [ISO-8601] of the record.

2.2.2.4 WINSINTF_ADD_VERS_MAP_T

The **WINSINTF_ADD_VERS_MAP_T** structure defines an **address version map** pair. This data structure is generally used by other data structures, such as **WINSINTF_RESULTS_T** and **WINSINTF_RESULTS_NEW_T**.

```
typedef struct _WINSINTF_ADD_VERS_MAP_T {
  WINSINTF_ADD_T Add;
  LARGE_INTEGER VersNo;
} WINSINTF_ADD_VERS_MAP_T,
 *PWINSINTF_ADD_VERS_MAP_T;
```

Add: A structure containing the IP address of a partner WINS server.

VersNo: The highest version number from all of the records owned by a WINS server at the target WINS server database. Each record in the database has a version number and owner Id associated with it.

2.2.2.5 WINSINTF_RPL_COUNTERS_T

The **WINSINTF_RPL_COUNTERS_T** structure defines counters that contain the number of successful pull replications and the number of communication failures for a given **replication partner**. It is used in the structure WINSINTF_STAT_T.

```
typedef struct _WINSINTF_RPL_COUNTERS_T {
  WINSINTF_ADD_T Add;
  DWORD NoOfRpls;
  DWORD NoOfCommFails;
} WINSINTF_RPL_COUNTERS_T,
 *PWINSINTF_RPL_COUNTERS_T;
```

Add: The IP address of a **partner** WINS server.

NoOfRpls: The number of successful pull replications that have been performed with the replication partner. The target WINS server stores the replication partner's IP address in the *Add* member.

NoOfCommFails: The number of communication failures that have occurred in pull replications between the WINS server whose IP address is given in *Add* and the target WINS server.

2.2.2.6 WINSINTF_STAT_T

The **WINSINTF_STAT_T** structure defines counters, configured timestamps, the pull replication statistics for a given WINS server. This structure is used by the structure **WINSINTF_RESULTS_T (section 2.2.2.7)**.

```
typedef struct _WINSINTF_STAT_T {
  struct {
    DWORD NoOfUniqueReg;
    DWORD NoOfGroupReg;
    DWORD NoOfQueries;
    DWORD NoOfSuccQueries;
    DWORD NoOfFailQueries;
    DWORD NoOfUniqueRef;
    DWORD NoOfGroupRef;
    DWORD NoOfRel;
    DWORD NoOfSuccRel;
    DWORD NoOfFailRel;
    DWORD NoOfUniqueCnf;
    DWORD NoOfGroupCnf;
  } Counters;
  struct {
    SYSTEMTIME WINSStartTime;
    SYSTEMTIME LastPScvTime;
    SYSTEMTIME LastATScvTime;
    SYSTEMTIME LastTombScvTime;
    SYSTEMTIME LastVerifyScvTime;
    SYSTEMTIME LastPRplTime;
    SYSTEMTIME LastATRplTime;
    SYSTEMTIME LastNTRplTime;
    SYSTEMTIME LastACTRplTime;
    SYSTEMTIME LastInitDbTime;
    SYSTEMTIME CounterResetTime;
  } TimeStamps;
  DWORD NoOfPnrs;
  [unique, size_is(NoOfPnrs)] PWINSINTF_RPL_COUNTERS_T pRplPnrs;
} WINSINTF_STAT_T,
 *PWINSINTF_STAT_T;
```

Counters: A structure that contains 32-bit unsigned integer counters, which measure various statistics on a WINS server.

NoOfUniqueReg: The number of unique registrations on the target WINS server since the service was started.

NoOfGroupReg: The number of group registrations at the target WINS server since the service was started.

NoOfQueries: The number of queries that clients have performed on the target WINS server to resolve NetBIOS names since the service was started. This value is the sum of the values maintained in *NoOfSuccQueries* and *NoOfFailQueries*.

NoOfSuccQueries: The number of successful **name_resolution** queries on the target WINS server since the service was started.

NoOfFailQueries: The number of failed name_resolution queries on the target WINS server since the service was started.

NoOfUniqueRef: The number of unique name refreshes on the target WINS server since the service was started.

NoOfGroupRef: The number of group name refreshes on the target WINS server since the service was started.

NoOfRel: The number of name releases on the target WINS server since the service was started. This value is the sum of the values maintained in *NoOfSuccRel* and *NoOfFailRel*.

NoOfSuccRel: The number of successful name releases on the target WINS server since the service was started.

NoOfFailRel: The number of failed name releases on the target WINS server since the service was started.

NoOfUniqueCnf: The number of unique name conflicts on the target WINS server since the service was started. Unique name conflicts can occur in the following cases:

- The server is registering or refreshing unique name requests from clients.
- The server is replicating unique name records from a partner WINS server.

NoOfGroupCnf: The number of group name conflicts on the target WINS server since the service was started. Group name conflicts can occur in the following cases:

- The server is registering or refreshing unique name requests from clients.
- The server is replicating unique name records from a partner WINS server.

TimeStamps: A structure that contains data in **SYSTEMTIME** structures ([MS-DTYP] section 2.3.11), which reflect the local time zone of the target WINS server.

WINSStartTime: The time at which the WINS service was started on the target WINS server.

LastPScvTime: The time at which the last periodic scavenging operation was done on the target WINS server.

LastATScvTime: The time at which the last administrator-triggered scavenging operation was done on the target WINS server.

LastTombScvTime: The time at which the last scavenging operation was done for the replicated tombstone records on the target WINS server.

LastVerifyScvTime: The time at which the last verification scavenging operation was done for the replicated active records on the target WINS server.

LastPRplTime: The time at which the last periodic pull replication was done on the target WINS server.

LastATRplTime: The time at which the last administrator-triggered pull replication was done on the target WINS server.

LastNTRplTime: This member is not set and MUST be ignored on receipt.

LastACTRplTime: This member is not set and MUST be ignored on receipt.

LastInitDbTime: The time at which the last static database initialization was done on the target WINS server.

CounterResetTime: The last time at which the administrator has cleared the success and failure replication counters of the target WINS server.

NoOfPnrs: The number of **pull partners** configured for the target WINS server.

pRplPnrs: A pointer to structures that contain the details of successful and failed replication counters of configured pull partners at the target WINS server, since the time service was started; or, the time at which the last reset happened by a call to the method **R_WinsResetCounters (section 3.1.4.12)**. The number of structures is specified by **NoOfPnrs**.

2.2.2.7 WINSINTF_RESULTS_T

The **WINSINTF_RESULTS_T** structure defines information related to the configuration and statistics of a target WINS server. This is used by RPC method **R_WinsStatus**.

```
typedef struct _WINSINTF_RESULTS_T {
  DWORD NoOfOwners;
  WINSINTF_ADD_VERS_MAP_T AddVersMaps[WINSINTF_MAX_NO_RPL_PNRS];
  LARGE_INTEGER MyMaxVersNo;
  DWORD RefreshInterval;
  DWORD TombstoneInterval;
  DWORD TombstoneTimeout;
  DWORD VerifyInterval;
  DWORD WINSPriorityClass;
  DWORD NoOfWorkerThds;
  WINSINTF_STAT_T WINSStat;
} WINSINTF_RESULTS_T,
 *PWINSINTF_RESULTS_T;
```

NoOfOwners: The number of owners whose records are part of the target WINS server database. The value of this member MUST be less than or equal to 25.

AddVersMaps: A structure containing the owner version map of the target WINS server. The number of valid entries is defined by the *NoOfOwners* value.

MyMaxVersNo: This member is not set and MUST be ignored on receipt.

RefreshInterval: The refresh time interval configured on the target WINS server, in seconds.

TombstoneInterval: The **tombstone time interval** configured on the target WINS server, in seconds.

TombstoneTimeout: The tombstone timeout configured on the target WINS server, in seconds.

VerifyInterval: The verify time interval configured on the target WINS server, in seconds.

WINSPriorityClass: The priority class of the WINS process running on the target WINS server. It SHOULD have one of the following values:<2>

Value	Meaning
NORMAL_PRIORITY_CLASS 0x00000020	The process has no special scheduling requirements.
HIGH_PRIORITY_CLASS 0x00000080	The process performs time-critical tasks that MUST be executed immediately for the process to run correctly. The threads of a high-priority class process preempt the threads of normal-priority class processes.

NoOfWorkerThds: The number of threads created in the WINS process for serving the NetBIOS name requests.

WINSStat: A **WINSINTF_STAT_T** structure (section 2.2.2.6) containing timing parameters configured on the target WINS server and the pull replication statistics of partner WINS servers.

2.2.2.8 WINSINTF_RECS_T

The structure **WINSINTF_RECS_T** defines an array of **WINSINTF_RECORD_ACTION_T (section 2.2.2.3)** elements. The **R_WinsGetDbRecs (section 3.1.4.6)** and **R_WinsGetDbRecsByName (section 3.1.4.19)** methods use this structure.

```
typedef struct _WINSINTF_RECS_T {
  DWORD BuffSize;
  [unique, size_is(NoOfRecs)] PWINSINTF_RECORD_ACTION_T pRow;
  DWORD NoOfRecs;
  DWORD TotalNoOfRecs;
} WINSINTF_RECS_T,
 *PWINSINTF_RECS_T;
```

BuffSize: The number of bytes allocated for the pointer *pRow*.

pRow: A pointer to an array of **WINSINTF_RECORD_ACTION_T** elements.

NoOfRecs: The number of records stored in the array pointed to by *pRow*.

TotalNoOfRecs: This member is not set and MUST be ignored on receipt.

2.2.2.9 WINSINTF_BROWSER_INFO_T

The **WINSINTF_BROWSER_INFO_T** structure defines information about browser names. It is used by the structure **WINSINTF_BROWSER_NAMES_T**.

```
typedef struct _WINSINTF_BROWSER_INFO_T {
  DWORD dwNameLen;
  [string] LPBYTE pName;
} WINSINTF_BROWSER_INFO_T,
 *PWINSINTF_BROWSER_INFO_T;
```

dwNameLen: The length of the name that **pName** points to, in bytes. This length includes the terminating null character.

pName: A pointer to a null-terminated string that contains the **browser name**.

2.2.2.10 WINSINTF_BROWSER_NAMES_T

The **WINSINTF_BROWSER_NAMES_T** structure defines an array of browser names. This structure is used by the RPC method **R_WinsGetBrowserNames**.

```
typedef struct _WINSINTF_BROWSER_NAMES_T {
  DWORD EntriesRead;
  [unique, size_is(EntriesRead)] PWINSINTF_BROWSER_INFO_T pInfo;
} WINSINTF_BROWSER_NAMES_T,
 *PWINSINTF_BROWSER_NAMES_T;
```

EntriesRead: The number of entries in the array that **pInfo** points to.

pInfo: A pointer to an array of browser names. **EntriesRead** contains the length of this array.

2.2.2.11 WINSINTF_RESULTS_NEW_T

The **WINSINTF_RESULTS_NEW_T** structure defines configuration information and statistics for a target WINS server. This structure is used by the RPC method **R_WinsStatusNew (section 3.1.4.20)**.

```
typedef struct _WINSINTF_RESULTS_NEW_T {
  DWORD NoOfOwners;
  [unique, size_is(NoOfOwners)] PWINSINTF_ADD_VERS_MAP_T pAddVersMaps;
  LARGE_INTEGER MyMaxVersNo;
  DWORD RefreshInterval;
  DWORD TombstoneInterval;
  DWORD TombstoneTimeout;
  DWORD VerifyInterval;
  DWORD WINSPriorityClass;
  DWORD NoOfWorkerThds;
  WINSINTF_STAT_T WINSStat;
} WINSINTF_RESULTS_NEW_T,
 *PWINSINTF_RESULTS_NEW_T;
```

NoOfOwners: The number of owners whose records are part of the target WINS server database.

pAddVersMaps: A pointer to an array of **WINSINTF_ADD_VERS_MAP_T** structure (section 2.2.2.4) elements. The **NoOfOwners** member contains the number of elements in the array.

MyMaxVersNo: This member is not set and MUST be ignored on receipt.

RefreshInterval: The refresh time interval configured on the target WINS server, in seconds.

TombstoneInterval: The tombstone time interval configured on the target WINS server, in seconds.

TombstoneTimeout: The tombstone timeout configured on the target WINS server, in seconds.

VerifyInterval: The verify time interval configured on the target WINS server, in seconds.

WINSPriorityClass: The priority class of the WINS process running on the target WINS server. It can have one of the following values:

Value	Meaning
NORMAL_PRIORITY_CLASS 0x00000020	The process has no special scheduling requirements.
HIGH_PRIORITY_CLASS 0x00000080	The process performs time-critical tasks that MUST be executed immediately for the process to run correctly. The threads of a high-priority class process preempt the threads of normal-priority class processes.

NoOfWorkerThds: The number of threads created in the WINS process to serve NetBIOS name requests.

WINSStat: A **WINSINTF_STAT_T** structure (section 2.2.2.6) containing timing parameters configured on the target WINS server and pull replication statistics of partner WINS servers.

2.2.2.12 WINSINTF_SCV_REQ_T

The **WINSINTF_SCV_REQ_T** structure defines the type of scavenging that needs to be done on the target WINS server. This is used by the RPC method **R_WinsDoScavengingNew (section 3.1.4.22)**.

```
typedef struct _WINSINTF_SCV_REQ_T {
  WINSINTF_SCV_OPC_E Opcode_e;
  DWORD Age;
  DWORD fForce;
} WINSINTF_SCV_REQ_T,
 *PWINSINTF_SCV_REQ_T;
```

Opcode_e: A **WINSINTF_SCV_OPC_E** enumeration (section 2.2.1.8) value describing the type of scavenging operation to be performed on the target WINS server.

Age: This member is not set and MUST be ignored on receipt.

fForce: Specifies whether a forceful scavenging is required.

Value	Meaning
0x00000000	The internal state and configuration of the WINS server determine whether scavenging is performed.

Value	Meaning
0x00000001 — 0xFFFFFFFF	The target WINS server performs scavenging.

3 Protocol Details

The client side of The Remote Administrative Interface: WINS protocol is simply a pass-through. This means that no additional timers or state are 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.

The WINS server supports two interfaces:

1. **winsif (section 3.1.4)**
2. **winsi2 (section 3.2.4)**

To support both interfaces, client applications are responsible for implementing mechanisms to manage memory such as the following:

- **midl_user_allocate**: Allocates memory for RPC input and output parameters.
- **midl_user_free**: Frees memory used by RPC input and output parameters.

Clients SHOULD call **midl_user_allocate** to allocate memory for any input pointer arguments to the RPC methods. The client RPC stub MUST send the input data to the server and then SHOULD free that memory by calling **midl_user_free**. Similarly, when the client RPC stub receives a response from server, it SHOULD call **midl_user_allocate** to allocate memory for all output pointer arguments. Client applications SHOULD free this memory by calling **midl_user_free**.

3.1 winsif Server Details

The methods supported by the **winsif** interface are specified in **Message Processing Events and Sequencing Rules (section 3.1.4)**.

3.1.1 Abstract Data Model

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

A **NetBIOS name server (NBNS)** maintains the following data structures:

Name record: A data structure that contains a name and the associated attributes.

Name records collection: A collection of all **name records** that are either registered by this NBNS server or obtained by replication.

Owner version map: A map between each NBNS owner and the record from that owner with the highest version number in the name records collection. This map determines whether the NBNS server pulls records from its **partners** and, if so, the range of records it obtains.

Global version counter: A 64-bit unsigned integer that tracks the version number given to the next record that is updated.

Server configuration: Parameters maintained in persistent storage include the following:

- Refresh interval

- **Extinction interval**
- Extinction timeout
- Verify interval
- Process priority class
- Number of worker threads

Browser name cache: A list of browser names that are stored in the target WINS server. When the WINS service is initialized, the cache SHOULD be empty.

The **Browser name cache** SHOULD be populated when the **R_WinsGetBrowserNames** method (section 3.1.4.18) is called for the first time, and every subsequent call to that method SHOULD return the contents of the cache. If 3 minutes or more has elapsed from the time the **Browser name cache** is refreshed, the WINS service SHOULD get the name records from the target WINS server database and update the cache.

3.1.2 Timers

No timers are required beyond those used internally by RPC to implement resiliency to network outages, as specified in [MS-RPCE] section 3.2.3.2.1.

3.1.3 Initialization

The **winsif** server for the Remote Administrative Interface: WINS MUST be initialized by registering the RPC interface and listening on the dynamically allocated port assigned by RPC, as specified in section 2.1. The client MUST contact the well-known RPC port on the WINS server to find the endpoint of **winsif**.

3.1.4 Message Processing Events and Sequencing Rules

The **winsif** interface provides methods that remotely configure, manage and monitor the WINS server.

Methods in RPC Opnum Order

Method	Description
R_WinsRecordAction	Inserts, modifies, deletes, releases or queries a Name Record from the WINS database. Opnum: 0
R_WinsStatus	Retrieves various counters, configuration settings and the statistics of a WINS server. Opnum: 1
R_WinsTrigger	Queues a request to trigger a replication from target WINS server to a specified WINS server. Opnum: 2
R_WinsDoStaticInit	Imports Name Records from a file to the WINS database. Opnum: 3

Method	Description
R_WinsDoScavenging	Queues a Scavenging request at the target WINS server. Opnum: 4
R_WinsGetDbRecs	Retrieves Name Records lying in a range of two version numbers and are owned by a particular WINS server. Opnum: 5
R_WinsTerm	Sends a termination signal to the WINS process running on the target WINS server. Opnum: 6
R_WinsBackup	Backs up the WINS database to a specified directory. Opnum: 7
R_WinsDelDbRecs	Deletes Name Records lying in a range of two version numbers and are owned by a particular WINS server. Opnum: 8
R_WinsPullRange	Pulls a range of records owned by a particular WINS server from another WINS server and replicates them with the target WINS server database. Opnum: 9
R_WinsSetPriorityClass	Modifies the priority class of a WINS process running on the target WINS server. Opnum: 10
R_WinsResetCounters	Resets all the pull replication partners counters stored at the target WINS server. Opnum: 11
R_WinsWorkerThdUpd	Modifies the number of NetBIOS threads to a new value at the target WINS server. Opnum: 12
R_WinsGetNameAndAdd	Retrieves the NetBIOS name and the corresponding IP address of the target WINS server. Opnum: 13
R_WinsGetBrowserNames_Old	This method SHOULD not be used. Opnum: 14
R_WinsDeleteWins	Deletes all the records owned by a particular WINS server from the target WINS server database. Opnum: 15
R_WinsSetFlags	This method SHOULD not be used. Opnum: 16
R_WinsGetBrowserNames	Retrieves the Browser Names information stored at the target WINS server. Opnum: 17

Method	Description
R_WinsGetDbRecsByName	Retrieves records matching a specified owner address from the target WINS server database. Opnum: 18
R_WinsStatusNew	Retrieves various configuration settings and the statistics of a WINS server. Opnum: 19
R_WinsStatusWHdl	Retrieves various configuration settings and the statistics of a WINS server. Opnum: 20
R_WinsDoScavengingNew	Requests a Scavenging operation at the target WINS server. Opnum: 21

3.1.4.1 R_WinsRecordAction (Opnum 0)

The **R_WinsRecordAction** method inserts, modifies, deletes, releases, or queries a name record from the WINS database.

```
DWORD R_WinsRecordAction(
  [in] handle_t ServerHdl,
  [in, out, ref] PWINSINTF_RECORD_ACTION_T* ppRecAction
);
```

ServerHdl: An RPC binding over IP address/**HostName** to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.<3>

ppRecAction: A pointer to a **WINSINTF_RECORD_ACTION_T** structure (section 2.2.2.3) that contains the details of the record and the action to be performed on it. The interpretation of the member values in this structure depends on the type of action specified by the **WINSINTF_ACT_E** enumeration (section 2.2.1.4) value in its **Cmd_e** member, as follows.

WINSINTF_E_INSERT:

- **Cmd_e** is set to WINSINTF_E_INSERT.
- **pName** points to a null-terminated string that contains the NetBIOS name and optionally the NetBIOS scope name of the record.
- **NameLen** contains the length of the string specified by **pName**.
- **TypOfRec_e** is set to a value between 0x00000000 and 0x00000003 based on the record type.
- **NoOfAdds** is set to a positive value based on the number of IP address mappings that the record has.
- **pAdd** or **Add** is set with the mapping IP addresses, based on the **TypOfRec_e** member.
- **VersNo** SHOULD be ignored by the server. The inserted record MUST be marked with the current version number that is in use at the WINS server.

- **NodeTyp** is set to a value between 0x00 and 0x03 based on the type of the node.
- **OwnerId** SHOULD be ignored by the server. The record MUST be inserted into the database with the **OwnerId** member set to the target WINS server address.
- **State_e** SHOULD be ignored by the server. The record MUST be inserted into the database with its state marked as **ACTIVE**.
- **fStatic** is set to 0x00000001 if the record being inserted is a static record; otherwise, it is set to 0x00000000.
- **TimeStamp** SHOULD be ignored by the server. The inserted record SHOULD be time-stamped with zero if the **fStatic** member is set to 0x00000001; otherwise, it SHOULD be time-stamped with the current time on the server plus the refresh interval configured on the server.

WINSINTF_E_DELETE:

- **Cmd_e** is set to WINSINTF_E_DELETE.
- **pName** points to a null-terminated string that contains the NetBIOS name and optionally the NetBIOS scope name of the record to be deleted from the database.
- **NameLen** contains the length of the string specified by **pName**.
- **State_e** is set to 0x00000003.
- All other members SHOULD be ignored by the server.

WINSINTF_E_RELEASE:

- **Cmd_e** is set to WINSINTF_E_RELEASE.
- **pName** points to a null-terminated string that contains the NetBIOS name and optionally the NetBIOS scope name of the record.
- **NameLen** contains the length of the string specified by **pName**.
- **TypOfRec_e** is set to a value between zero and 0x00000003 based on the record type.
- **NoOfAdds** MUST be set to 0x00000001.
- **pAdd** or **Add** is set with the mapping IP address based on the **TypOfRec_e** member.
- **VersNo**, **NodeTyp**, **OwnerId**, and **fStatic** SHOULD be ignored by the server.
- **State_e** SHOULD be ignored by the server. The record MUST be inserted with state marked as **RELEASED**.
- **TimeStamp** SHOULD be ignored by the server. The released record SHOULD be time-stamped with 0xFFFFFFFF if the **fStatic** member is set to 0x00000001; otherwise, it SHOULD be time-stamped with the current time on the server plus the tombstone interval configured on the server.

WINSINTF_E_MODIFY:

- **Cmd_e** is set to WINSINTF_E_MODIFY.

- **pName** points to a null-terminated string that contains the NetBIOS name and optionally the NetBIOS scope name of the record to be modified in the database.
- **NameLen** contains the length of the string specified by **pName**.
- **TypOfRec_e** contains the record type to be set for the record matching the **pName** member in the WINS database.
- **NodeTyp** contains the node type to be set for the record matching the **pName** member in the WINS database.
- **State_e** contains the state to be set for the record matching the **pName** member in the WINS database.
- **fStatic** contains the value to be set for the record matching the **pName** member in the WINS database.
- All other members SHOULD be ignored by the server.

WINSINTF_E_QUERY:

- **Cmd_e** is set to WINSINTF_E_QUERY.
- **pName** points to a null-terminated string that contains the NetBIOS name and optionally the NetBIOS scope name of the record to be queried from the database.
- **NameLen** contains the length of the string specified by **pName**.
- All other members act as output, which are filled by the server if a matching entry is found in the database.
- **TypOfRec_e** contains the matching record type.
- If the **TyeOfRec_e** member is set to 0x00000000 or 0x00000001, the **NoOfAdds** member SHOULD contain 0x00000001 or the number of IP addresses that are mapped to the name given in the **pName** member.
- If the **TypOfRec_e** member is set to 0x00000002 or 0x00000003. The RPC method caller SHOULD refer to this member for the set of IP addresses mapped to the name given in the **pName** member.
- If the **TypOfRec_e** member is set to 0x00000000 or 0x00000001. The RPC method caller SHOULD refer to this member for the IP address mapped to the name given in the **pName** member. If the **TypOfRec_e** member is set to 0x00000001, the **IPAdd** member of the **Add** structure MUST contain 0xFFFFFFFF.
- **VersNo** contains the version number of the matching record.
- **NodeTyp** contains the node type of the matching record.
- **OwnerId** contains the IP address of the owner of the matching record.
- **State_e** contains the state of the matching record.
- **fStatic** contains the value 0x00000001 if the record is entered into the database by an administrator; otherwise, it contains 0x00000000.
- **TimeStamp** contains the time stamp of the record.

Return Values: A 32-bit unsigned integer value that indicates return status. A return value of ERROR_SUCCESS indicates that the operation was completed successfully. Otherwise, the **TimeStamp** member SHOULD contain one of the following Win32 error codes, as specified in [MS-ERREF]:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.
0x00000FA5 ERROR_REC_NON_EXISTENT	The name does not exist in the database. This error is returned only if a requested WINSINTF_E_QUERY operation is not successful.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

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

Processing and Response Requirements:

The **Opnum** value for this method is 0.

When processing this call, the WINS server MUST do the following:

- If the action specified is WINSINTF_E_QUERY, the RPC method caller SHOULD have query level access.<4> For all other actions the caller SHOULD have control level access. If an RPC client with a lower access level calls this method, the server SHOULD return ERROR_ACCESS_DENIED.
- The WINS service on the target WINS server MUST be in the running or paused state for this method to complete successfully. If the service is in initializing or exiting state, ERROR_WINS_INTERNAL status SHOULD be returned.
- When the RPC method is called with an action set to WINSINTF_E_INSERT, the requested record is inserted into the database. If the record with the same name already exists in the database, name resolution occurs as described in [MS-WINSRA]. The server returns ERROR_WINS_INTERNAL, if any error occurs while performing the resolution or inserting the record.<5>
- When an RPC method is called with the action set to WINSINTF_E_RELEASE, the state of the matching record is changed to RELEASED in the database. If a matching record is not found, the server returns ERROR_SUCCESS. If any failure occurs during the modification of the record state, ERROR_WINS_INTERNAL is returned.
- When an RPC method is called with the action set to WINSINTF_E_MODIFY, the database is searched for a matching record. If a match is found, the attributes of the record such as record type, node type, record state, and *fstatic* are modified according to the requested values. If the matching record's type is either unique or Normal Group and a request comes to modify it to multihomed or Special Group, respectively, an ERROR_WINS_INTERNAL error is returned; otherwise, ERROR_SUCCESS is returned. If the record is not found in the database, the server returns ERROR_SUCCESS.

- When the RPC method is called with the action set to WINSINTF_E_QUERY, the database is queried for the given name. If a matching record is found, the attributes of the record are returned to the RPC caller. If the record is not found or if any error occurs during attribute retrieval, the server returns an ERROR_REC_NON_EXISTENT error.
- When the RPC method is called with the action set to WINSINTF_E_DELETE, the matching record is deleted from the database. If a matching record is not found in the database, an ERROR_SUCCESS status code is returned. If any error occurs during the database operations, an ERROR_WINS_INTERNAL is returned. The RPC method caller MUST set *state_e* to DELETED for this action to succeed.

3.1.4.2 R_WinsStatus (Opnum 1)

The **R_WinsStatus** method retrieves configuration settings and statistics from a WINS server.

```
DWORD R_WinsStatus(
  [in] handle_t ServerHdl,
  [in] WINSINTF_CMD_E Cmd_e,
  [in, out, ref] PWINSINTF_RESULTS_T pResults
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

Cmd_e: The command to be executed on the target WINS server from the **WINSINTF_CMD_E** enumeration (section 2.2.1.5).

pResults: A pointer to a **WINSINTF_RESULTS_T** structure (section 2.2.2.7) that contains configuration data and statistics for the target WINS server.

Return Values: A 32-bit unsigned integer value that indicates return status. A return value of ERROR_SUCCESS (0x00000000) indicates that the operation completed successfully. Otherwise, this return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned.

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

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

Processing and Response Requirements:

The following requirements and recommendations apply to the WINS server that processes a call to **R_WinStatus**:

- The **R_WinStatus** caller SHOULD have query level access.<6> If an RPC client with a lower access level calls **R_WinStatus**, the server SHOULD return ERROR_ACCESS_DENIED.

- The WINS service on the target WINS server MUST be in the running or paused state. If the service is in initializing or exiting state, the server SHOULD return ERROR_WINS_INTERNAL.
- When **R_WinStatus** is called with the **Cmd_e** parameter set to WINSINTF_E_ADDVERSMAP, the first entry of the *AddVersMaps* array SHOULD contain the address of the WINS server for which the version number is requested. The WINSINTF_E_ADDVERSMAP command retrieves the version number for the specified owner address from the owner version map of the target WINS server. The retrieved version number is stored in the *AddVersMaps[0].VersNo* parameter. If the address of the **owner WINS server** is not found in the owner-version map of the target WINS server, an ERROR_WINS_INTERNAL error is returned.
- When **R_WinStatus** is called with the *Cmd_e* parameter set to WINSINTF_E_CONFIG, the *pResults* parameter is used only for output. The *NoOfOwners* and *AddVersMaps* parameters specify the owner version map table maintained on the target WINS server. If the owner version map table has more than 25 entries, only the first 25 entries are copied to *pResults->AddVersMaps*. The *RefreshInterval*, *TombstoneInterval*, *TombstoneTimeout*, *VerifyInterval*, *WINSPriorityClass*, and *NoOfWorkerThds* members get values according to the configuration of the target WINS server. The *WINSStat* parameter is not used for this command. An ERROR_WINS_INTERNAL error is returned if any error occurs during processing of a WINSINTF_E_CONFIG command.
- When **R_WinStatus** is called with the **Cmd_e** parameter set to WINSINTF_E_CONFIG_ALL_MAPS, the behavior is the same as specified for the WINSINTF_E_CONFIG command except that the owner version map entry is returned even if it is marked as deleted.
- When **R_WinStatus** is called with the **Cmd_e** parameter set to WINSINTF_E_STAT, the *pResults* parameter is used only for output. Statistics for the target WINS server are copied to *pResults->WINSStat*. The *pResults->WINSStat.pRplPnrs* pointer MUST be NULL for this operation to succeed. The WINSINTF_E_STAT command also retrieves the information retrieved by the WINSINTF_E_CONFIG command. An ERROR_WINS_INTERNAL error is returned if any error occurs during processing of a WINSINTF_E_STAT command.

3.1.4.3 R_WinsTrigger (Opnum 2)

The **R_WinsTrigger** method triggers a replication operation between a target WINS server and another WINS server.

```
DWORD R_WinsTrigger(
  [in] handle_t ServerHdl,
  [in, ref] PWINSINTF_ADD_T pWinsAdd,
  [in] WINSINTF_TRIG_TYPE_E TrigType_e
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

pWinsAdd: Address of the WINS server with which the target WINS server performs the replication operation.

TrigType_e: The type of replication operation requested.

Return Values: A 32 bit unsigned integer value that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that the operation completed successfully. Any

other return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.
0x00000FA6 ERROR_RPL_NOT_ALLOWED	The WINS server requested for the replication operation is requested is not configured as a replication partner for the target WINS server.
0x00000005 ERROR_ACCESS_DENIED	The caller does not have sufficient permissions.

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

Processing and Response Requirements:

When **R_WinsTrigger** is called, the server returns immediately without waiting for the replication operation to finish. The server just queues a request for the replication operation, and the replication takes place at a time determined by the internal state and configuration of the target WINS server. Hence, applications that call **R_WinsTrigger** SHOULD NOT treat an ERROR_SUCCESS return value as indicating a successful replication operation. Instead, applications SHOULD rely on WINS event logs to determine whether or not replication is successful.

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsTrigger**:

- The **R_WinsTrigger** caller SHOULD have control level access. If an RPC client with a lower access level calls **R_WinsTrigger**, the server SHOULD return ERROR_ACCESS_DENIED.
- The WINS service on the target WINS server MUST be in the running or paused state. If the service is in initializing or exiting state, the server SHOULD return ERROR_WINS_INTERNAL for its status.
- When **R_WinsTrigger** is called with trigger type WINSINTF_E_PUSH, the server queues a push replication. If the target WINS server is configured to replicate only with partners and the address of the requested replication partner is not in the server's list of push replication partners, the server SHOULD return WINSINTF_RPL_NOT_ALLOWED. The server can also return ERROR_WINS_INTERNAL for any other errors occur while it processes the request.
- The trigger type WINSINTF_E_PUSH_PROP works same way as the command WINSINTF_E_PUSH except that the update notifications that are sent as part of push replication have the propagate opcode set [MS-WINSRA].
- When the **R_WinsTrigger** method is called with trigger type WINSINTF_E_PULL, the server queues a pull replication as specified by the *pWinsAdd* parameter. If the target WINS server is configured to replicate only with partners, and the address of the requested replication partner is not in the server's list of pull replication partners, the server SHOULD return WINSINTF_RPL_NOT_ALLOWED. Also, the server SHOULD return ERROR_WINS_INTERNAL for any other errors that occur while it processes the request.

3.1.4.4 R_WinsDoStaticInit (Opnum 3)

The **R_WinsDoStaticInit** method performs static initialization of a WINS database by registering the names specified in a data file.

```
DWORD R_WinsDoStaticInit(
  [in] handle_t ServerHdl,
  [in, unique, string] LPWSTR pDataFilePath,
  [in] DWORD fDel
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to resolve which WINS server the call is directed to.

pDataFilePath: A pointer to a **Unicode string** containing the path to a text file on the target WINS server. The file SHOULD contain entries that map NetBIOS names to **IPv4 addresses in string format** using the following syntax:

```
<IPv4 address 1> <one or more spaces> <NetBIOS name 1>
<IPv4 address 2> <one or more spaces> <NetBIOS name 2>
...
<IPv4 address N> <one or more spaces> <NetBIOS name N>
```

An example of this syntax can be found in the Windows **LMHOSTS** file. See [LMHOSTS] for more information.

If this pointer value is NULL, the target WINS server SHOULD use the following default path: "%systemroot%\system32\drivers\etc\lmhosts".

fDel: Value specifying whether or not to delete the file specified by *pDataFilePath* from the target WINS server. A non-zero value deletes the file from the target WINS server after the database initialization is complete.

Return Values: A 32 bit unsigned integer value that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that the operation completed successfully. Any other return value is a Win32 error code as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000FA2 ERROR_STATIC_INIT_FAILED	An error occurred during static initialization of the database file.
0x00000005 ERROR_ACCESS_DENIED	The caller does not have sufficient permissions.

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

Processing and Response Requirements:

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsDoStaticInit**:

- The **R_WinsDoStaticInit** caller SHOULD have control level access. If an RPC client with a lower access level calls this method, the server SHOULD return ERROR_ACCESS_DENIED.
- The WINS server retrieves the entries from the specified file and registers the retrieved names into the WINS database.
- After the WINS server finishes the initialization, it removes the file if *fDel* is set to a nonzero value.
- The WINS server SHOULD return ERROR_STATIC_INIT_FAILED if any error occurs while the server is reading the file or registering the names in the database.

3.1.4.5 R_WinsDoScavenging (Opnum 4)

The **R_WinsDoScavenging** method queues a scavenging request on the target WINS server.

```
DWORD R_WinsDoScavenging(
  [in] handle_t ServerHdl
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

Return Values: A 32 bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that the operation completed successfully. A nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

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

Processing and Response Requirements:

When **R_WinsDoScanvenging** is called, the server returns immediately without waiting for scavenging to start. The server just queues a request for the scavenging operation, and the internal state and configuration of the WINS server determine whether or not the scavenging occurs. Hence,callers to **R_WinsDoScanvenging** SHOULD NOT treat a return code of ERROR_SUCCESS as indicating a successful scavenging operation. Instead, callers SHOULD rely on WINS event logs to determine whether or not the scavenging operation succeeded.

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsDoScavenging**:

- Callers to **R_WinsDoScavenging** SHOULD have control level access. If an RPC client with a lower access level calls **R_WinsDoScavenging**, the server SHOULD return ERROR_ACCESS_DENIED.
- The WINS server queues a request on the target WINS server for the scavenging operation, and the method returns immediately with ERROR_SUCCESS as the status code.

3.1.4.6 R_WinsGetDbRecs (Opnum 5)

The **R_WinsGetDbRecs** method returns the records whose version numbers are within a specified range and that are owned by a specified WINS server.

```
DWORD R_WinsGetDbRecs(
  [in] handle_t ServerHdl,
  [in, ref] PWINSINTF_ADD_T pWinsAdd,
  [in] WINSINTF_VERS_NO_T MinVersNo,
  [in] WINSINTF_VERS_NO_T MaxVersNo,
  [out] PWINSINTF_RECS_T pRecs
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding to resolve which WINS server the call is directed to.

pWinsAdd: Address of an owner WINS server whose records are retrieved from the target WINS server.

MinVersNo: The lower bound on the version range of the records to be retrieved.

MaxVersNo: The upper bound on the version range of the records to be retrieved.

pRecs: Pointer to a structure of type **WINSINTF_RECS_T**, which contains the records retrieved from the target WINS server.

Return Values: A 32-bit unsigned integer value that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that the operation finished successfully. Any nonzero value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.
0x00000005 ERROR_ACCESS_DENIED	The caller does not have sufficient permissions.

Exceptions Thrown: No exceptions SHOULD be thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

Processing and Response Requirements:

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsGetDbRecs**:

- The RPC method caller SHOULD have query-level access.<7> If an RPC client with a lower access level calls **R_WinsGetDbRecs**, the server SHOULD return ERROR_ACCESS_DENIED.
- In response to a **R_WinsGetDbRecs** call, records are retrieved from the target WINS server database if their version numbers fall between *MinVersNo* and *MaxVersNo*, and if the records are owned by the owner WINS server whose address is specified by *pWinsAdd*.
- If the **R_WinsGetDbRecs** caller specifies zero for both *MinVersNo* and *MaxVersNo*, all records owned by the WINS server specified by *pWinsAdd* are retrieved from the target WINS server's database.
- The *MinVersNo* value MUST be less than or equal to *MaxVersNo* value for the **R_WinsGetDbRecs** call to succeed; otherwise, the server SHOULD return ERROR_WINS_INTERNAL.

The **R_WinsGetDbRecs** caller is responsible for freeing the memory pointed to by *pRecs->pRow->pName* and *pRecs->pRow->pAdd* for each record, then using the **midl_user_free** function (section 3) to free the *pRecs->pRow* and *pRecs* pointers themselves.

3.1.4.7 R_WinsTerm (Opnum 6)

The **R_WinsTerm** method sends a termination signal to the WINS process on a target WINS server.

```
DWORD R_WinsTerm(
  [in] handle_t ServerHdl,
  [in] SHORT fAbruptTem
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

fAbruptTem: A value that indicates whether the WINS process terminates immediately. If this value is nonzero, the service terminates immediately. Otherwise, the service exits normally and frees all resources.

Return Values: A 32 bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that operation completed successfully. A nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

Exceptions Thrown: No exceptions SHOULD be thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

Processing and Response Requirements:

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsTerm**:

- The **R_WinsTerm** caller SHOULD have control level access. If an RPC client with a lower access level calls **R_WinsTerm**, the server SHOULD return ERROR_ACCESS_DENIED.
- **R_WinsTerm** always returns ERROR_SUCCESS if the client has sufficient access level permissions.
- If *fAbruptTem* is set to a nonzero value, the service exits immediately. Otherwise, the service frees all the resources and then calls the exit process.

3.1.4.8 R_WinsBackup (Opnum 7)

The **R_WinsBackup** method backs up the WINS database to a specified directory.

```
DWORD R_WinsBackup(
  [in] handle_t ServerHdl,
  [in, string, ref] LPBYTE pBackupPath,
  [in] SHORT fIncremental
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

pBackupPath: A pointer to a string that contains the name of the directory to which to back up the database. This pointer MUST not be NULL.

fIncremental: A value that is ignored.

Return Values: A 32-bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that the operation completed successfully. Any nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned.

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.
0x00000FA4 ERROR_FULL_BACKUP	The backup failed. Check the directory to which you are backing up the database.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

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

Processing and Response Requirements:

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsBackup**:

- The **R_WinsBackup** caller SHOULD have control level access. If an RPC client with a lower access level calls this method, the server SHOULD return ERROR_ACCESS_DENIED.

- The server returns ERROR_WINS_INTERNAL if *pBackupPath* points to a string that is longer than 255 characters.
- The database is always backed up to the path specified by *pBackupPath* with the string "\wins_bak\" appended. If the client doesn't have sufficient permissions to create files in the specified directory or if the backup fails for any other reasons, the server SHOULD return an ERROR_FULL_BACKUP error.

3.1.4.9 R_WinsDelDbRecs (Opnum 8)

The **R_WinsDelDbRecs** method deletes the records whose version numbers are within a specified range and that are owned by a specified WINS server.

```
DWORD R_WinsDelDbRecs(
  [in] handle_t ServerHdl,
  [in, ref] PWINSINTF_ADD_T pWinsAdd,
  [in] WINSINTF_VERS_NO_T MinVersNo,
  [in] WINSINTF_VERS_NO_T MaxVersNo
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

pWinsAdd: A pointer to an owner WINS server address whose records are to be deleted from the target WINS server.

MinVersNo: The lower bound on the version number of the records to be deleted.

MaxVersNo: The upper bound on the version number of the records to be deleted.

Return Values: A 32-bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that the operation completed successfully. Any nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned.

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

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

Processing and Response Requirements:

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsDelDbRecs**:

- The **R_WinsDelDbRecs** caller SHOULD have control-level access. If an RPC client with a lower access level calls this method, the server SHOULD return ERROR_ACCESS_DENIED.

- If the target WINS server doesn't have any records owned by the WINS server whose address is specified by *pWinsAdd*, the server SHOULD return ERROR_WINS_INTERNAL.
- Records are deleted from the target WINS server database if their version numbers fall between the values of *MinVersNo* and *MaxVersNo* and if they are owned by the WINS server whose address is specified in *pWinsAdd*.
- If both *MinVersNo* and *MaxVersNo* are set to zero, all records owned by the WINS server whose address is specified in *pWinsAdd* are deleted.

3.1.4.10 R_WinsPullRange (Opnum 9)

The **R_WinsPullRange** method pulls a range of records owned by a WINS server from another WINS server, and replicates them within the target WINS server database.<8>

```
DWORD R_WinsPullRange(
  [in] handle_t ServerHdl,
  [in, ref] PWINSINTF_ADD_T pWinsAdd,
  [in, ref] PWINSINTF_ADD_T pOwnerAdd,
  [in] WINSINTF_VERS_NO_T MinVersNo,
  [in] WINSINTF_VERS_NO_T MaxVersNo
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

pWinsAdd: The address of the WINS server from which the entries are pulled.

pOwnerAdd: The address of the owner WINS server whose entries are pulled.

MinVersNo: The lower bound on the range of version numbers for the records to be pulled.

MaxVersNo: The upper bound on the range of version numbers for the records to be pulled.

Return Values: A 32 bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that operation completed successfully. A nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

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

Processing and Response Requirements:

When **R_WinsPullRange** is called, the server returns immediately without waiting for the actual pull. It just queues a request for the pull operation, and the actual pull starts at a time determined by the current state and configuration of the target WINS server. Hence, **R_WinsPullRange** callers SHOULD NOT treat an ERROR_SUCCESS return value as indicating a successful pull operation. Instead, callers SHOULD rely on WINS event logs to determine whether or not the pull operation succeeded.

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsPullRange**:

- **R_WinsPullRange** callers SHOULD have control level access. If an RPC client with a lower access level calls this method, the server SHOULD return ERROR_ACCESS_DENIED.
- The value of *MinVersNo* MUST be less than or equal to the value of *MaxVersNo*. Otherwise, the server SHOULD return ERROR_WINS_INTERNAL.
- If the target WINS server is configured to pull records only from configured partners, the WINS server address given in *pWinsAdd* MUST have been configured as a pull partner for the target WINS server. Otherwise, the server SHOULD return ERROR_WINS_INTERNAL.
- When the client queues a request to pull the records owned by the server whose address is given in *pOwnerAdd* from the WINS server whose address is given in *pWinsAdd*, the RPC call SHOULD return immediately without waiting for the replication operation to complete.

3.1.4.11 R_WinsSetPriorityClass (Opnum 10)

The **R_WinsSetPriorityClass** method sets the priority class for the WINS process running on the target WINS server.

```
DWORD R_WinsSetPriorityClass(
  [in] handle_t ServerHdl,
  [in] WINSINTF_PRIORITY_CLASS_E PrCls_e
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

PrCls_e: The priority class to be set.

Return Values: A 32 bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that the operation completed successfully. A nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

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

Processing and Response Requirements:

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsSetPriorityClass**:

- The **R_WinsSetPriorityClass** caller SHOULD have control level access. If an RPC client with a lower access level calls this method, the server SHOULD return ERROR_ACCESS_DENIED.
- If *PrCls_e* is set to a value other than WINSINTF_E_NORMAL or WINSINTF_E_HIGH, the server SHOULD return ERROR_WINS_INTERNAL.
- The server sets the priority class of the WINS process to the one specified by *PrsCls_e*.

3.1.4.12 R_WinsResetCounters (Opnum 11)

The **R_WinsResetCounters** method resets the pull replication counters for all partners of the target WINS server.

```
DWORD R_WinsResetCounters(
  [in] handle_t ServerHdl
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

Return Values: A 32 bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that operation completed successfully. A nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

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

Processing and Response Requirements:

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsResetCounters**:

- The **R_WinsResetCounters** caller SHOULD have control level access. If an RPC client with a lower access level calls this method, the server SHOULD return ERROR_ACCESS_DENIED.
- Each WINS server maintains one **WINSINTF_RPL_COUNTERS_T** structure (section 2.2.2.5) per configured pull partner to track the number of successful pull replications and the number of communication failures. The **R_WinsResetCounters** method resets to zero the values of the

NoOfRpls and **NoOfCommFails** members of the **WINSINTF_RPL_COUNTERS_T** structures for all the configured pull partners of the target WINS server.

- This method MUST return ERROR_SUCCESS if the client has sufficient access level permissions.

3.1.4.13 R_WinsWorkerThdUpd (Opnum 12)

The **R_WinsWorkerThdUpd** method updates the number of threads that have been created to serve NetBIOS requests.

```
DWORD R_WinsWorkerThdUpd(
  [in] handle_t ServerHdl,
  [in] DWORD NewNoOfNbtThds
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

NewNoOfNbtThds: New value for the number of worker threads that have been created for NetBIOS requests.

Return Values: A 32 bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that operation completed successfully. A nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

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

Processing and Response Requirements:

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsWorkerThdUpd**:

- The **R_WinsWorkerThdUpd** caller SHOULD have control level access. If an RPC client with a lower access level calls this method, the server SHOULD return ERROR_ACCESS_DENIED.
- The WINS service MUST be in the running or paused state for this method to succeed. If the service is in the initializing or exiting state, the server SHOULD return ERROR_WINS_INTERNAL.
- The new number given in *NewNoOfNbtThds* MUST be in the range 2 through 19, inclusive. Otherwise, the server SHOULD return an ERROR_WINS_INTERNAL error.
- The **R_WinsWorkerThdUpd** call sets the number of worker threads that serve NetBIOS requests to the new number given in *NewNoOfNbtThds*. If the existing number of NetBIOS

threads is same as the requested number, the RPC call SHOULD return immediately. Otherwise, NetBIOS threads are created or deleted to adjust the total number of threads to the requested number.

3.1.4.14 R_WinsGetNameAndAdd (Opnum 13)

The **R_WinsGetNameAndAdd** method retrieves the NetBIOS name and the corresponding IP address of the target WINS server.

```
DWORD R_WinsGetNameAndAdd(
  [in] handle_t ServerHdl,
  [out, ref] PWINSINTF_ADD_T pWinsAdd,
  [out, string, size_is(80)] LPBYTE pUncName
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

pWinsAdd: A pointer to a structure containing the IP address of the target WINS server.

pUncName: A pointer to a null-terminated string containing the NetBIOS name of the target WINS server.

Return Values: A 32 bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that operation completed successfully. A nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

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

Processing and Response Requirements:

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsGetNameAndAdd**:

- The **R_WinsGetNameAndAdd** caller SHOULD have query-level access.<9> If an RPC client with a lower access level calls this method, the server SHOULD return ERROR_ACCESS_DENIED.<10>
- The structure that *pWinsAdd* points to contains only an IP address. The **R_WinsGetNameAndAdd** caller SHOULD ignore the other fields of the structure.
- The server retrieves the NetBIOS name by calling a standard Windows function, which returns the status code directly to the caller without any modification. Hence, any Win32 error code can be returned, as specified in [MS-ERREF].

3.1.4.15 R_WinsGetBrowserNames_Old (Opnum 14)

The **R_WinsGetBrowserNames_Old** method always returns an ERROR_WINS_INTERNAL error code.

```
DWORD R_WinsGetBrowserNames_Old(
  [in] handle_t ServerHdl,
  [out] PWINSINTF_BROWSER_NAMES_T pNames
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

pNames: This field MUST be ignored.

Return Values: A 32-bit unsigned integer value that indicates the return status. The method always returns the ERROR_WINS_INTERNAL error code.

Return value/code	Description
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.

Exceptions Thrown: No exceptions are thrown.

Processing and Response Requirements:

Clients with any access level can call this method.

3.1.4.16 R_WinsDeleteWins (Opnum 15)

The **R_WinsDeleteWins** method deletes all the records owned by a particular WINS server from the target WINS server database.

```
DWORD R_WinsDeleteWins(
  [in] handle_t ServerHdl,
  [in, ref] PWINSINTF_ADD_T pWinsAdd
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

pWinsAdd: A pointer to the address of the owner WINS server whose records are to be deleted from the target WINS server.

Return Values: A 32 bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that operation completed successfully. A nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.

Return value/code	Description
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

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

Processing and Response Requirements:

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsDeleteWins**:

- The RPC method caller SHOULD have control-level access. If an RPC client with a lower access level calls this method, the server SHOULD return ERROR_ACCESS_DENIED.
- If *pWinsAdd* contains the IP address of the target WINS, the records are deleted immediately from the target WINS server database. If the server encounters an error while retrieving the records from the database, it SHOULD return ERROR_WINS_INTERNAL; otherwise, the server returns ERROR_SUCCESS.
- If *pWinsAdd* contains an IP address different from the target WINS server address, a request is queued at the target WINS server, and the RPC call returns immediately with ERROR_SUCCESS status.

3.1.4.17 R_WinsSetFlags (Opnum 16)

The **R_WinsSetFlags** method always returns ERROR_SUCCESS.

```
DWORD R_WinsSetFlags(
  [in] handle_t ServerHdl,
  [in] DWORD fFlags
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

fFlags: This field MUST be ignored.

Return Values: A 32-bit unsigned integer value that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that the operation completed successfully.

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.

Exceptions Thrown: No exceptions are thrown.

Processing and Response Requirements:

Clients with any access level can call this method.

3.1.4.18 R_WinsGetBrowserNames (Opnum 17)

The **R_WinsGetBrowserNames** method retrieves browser name records from the target WINS server database.

```
DWORD R_WinsGetBrowserNames(
  [in, ref] WINSIF_HANDLE ServerHdl,
  [out] PWINSINTF_BROWSER_NAMES_T pNames
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to. See [MSDN-Handles] for more information.

This value MUST be ignored by the WINS server on receipt.

pNames: A pointer to a structure of type **WINSINTF_BROWSER_NAMES_T (section 2.2.2.10)**, which contains the browser name records retrieved from the target WINS server.

Return Values: A 32-bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that operation completed successfully. A nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.

Exceptions Thrown: No exceptions SHOULD be thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

Processing and Response Requirements:

Clients with any access level can call this method.

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsGetBrowserNames**:

- This method retrieves all browser name records in the target WINS server database.
- If the **Browser name cache** abstract data element (section 3.1.1) has been populated, and less than 3 minutes have elapsed since it was last updated, this method SHOULD return the records from the cache by using the *pNames* parameter.
- If this method call is being made for the first time, or if 3 minutes or more have elapsed since the **Browser name cache** was last updated, the cache SHOULD be refreshed by fetching records from the database, and the contents of the cache are returned.
- If any error occurs while retrieving the records, the service SHOULD return an ERROR_WINS_INTERNAL error code.

The **R_WinsGetBrowserNames** caller is responsible for freeing the memory pointed to by *pRecs->pRow->pName* and *pRecs->pRow->pAdd* for each record, then using the **midl_user_free** function (section 3) to free the *pRecs->pRow* and *pRecs* pointers themselves.

3.1.4.19 R_WinsGetDbRecsByName (Opnum 18)

The **R_WinsGetDbRecsByName** method retrieves records matching an owner address from a target WINS server database starting at a specified cursor.

```
DWORD R_WinsGetDbRecsByName(
  [in] handle_t ServerHdl,
  [in, unique] PWINSINTF_ADD_T pWinsAdd,
  [in] DWORD Location,
  [in, unique, size_is(NameLen + 1)]
    LPBYTE pName,
  [in] DWORD NameLen,
  [in] DWORD NoOfRecsDesired,
  [in] DWORD fOnlyStatic,
  [out] PWINSINTF_RECS_T pRecs
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

pWinsAdd: A pointer to the address of the owner WINS server whose records are to be retrieved. If the pointer is NULL, the records for all owners are retrieved.

Location: A value specifying the direction in which the database is searched. If the value is zero, the database is searched forward starting from the beginning. If the value is 1, the database is searched backward starting from the last record of the database.

pName: A pointer to a name that specifies the cursor from which the database retrieval starts.

NameLen: The length of the name that *pName* points to, including terminating null character.

NoOfRecsDesired: The number of records to be retrieved from the database.

fOnlyStatic: Takes a value of 1, 2, or 4 to indicate whether static records, dynamic records, or both are retrieved. A value of 1 retrieves only static records. A value of 2 retrieves only dynamic records. A value of 4 retrieves both static records and dynamic records.

pRecs: A pointer to a structure containing the retrieved records.

Return Values: A 32 bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that operation completed successfully. A nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.

Return value/code	Description
0x00000FA5 ERROR_REC_NON_EXISTENT	No records were found matching the given data.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

Exceptions Thrown: No exceptions SHOULD be thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

Processing and Response Requirements:

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsGetDbRecsByName**:

- The RPC method caller SHOULD have query-level access.<11> If an RPC client with a lower access level calls this method, the server SHOULD return ERROR_ACCESS_DENIED.
- This method returns all records whose owner address matches the address specified in *pWinsAdd*. If *pName* points to a valid name, the database search starts from the record after the record whose name matches the valid name. If the name that *pName* points to does not match the name for any database records, the database search starts from the beginning of the database.
- A maximum of 5,000 records can be retrieved in a single call.
- If the owner's address is specified and if the server can't find this address in its owner version map, the server returns error ERROR_WINS_INTERNAL error.
- If no records match the search criteria, the server returns an ERROR_REC_NON_EXISTENT error. For any other error conditions, the server returns an ERROR_WINS_INTERNAL error.
- Refer to Retrieving All the Records of a WINS Database (section 4.6) to see how to use **R_WinsGetDbRecsByName** to retrieve all the records of a database.

The **R_WinsGetDbRecsByName** caller is responsible for freeing the memory pointed to by *pRecs->pRow->pName* and *pRecs->pRow->pAdd* for each record, then using the **midl_user_free** function (section 3) to free the *pRecs->pRow* and *pRecs* pointers themselves.

3.1.4.20 R_WinsStatusNew (Opnum 19)

The **R_WinsStatusNew** method retrieves configuration settings and statistics from a WINS server.

```
DWORD R_WinsStatusNew(
  [in] handle_t ServerHdl,
  [in] WINSINTF_CMD_E Cmd_e,
  [out] PWINSINTF_RESULTS_NEW_T pResults
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

Cmd_e: The command to be executed on the target WINS server, from the **WINSINTF_CMD_E** enumeration (section 2.2.1.5).

pResults: A pointer to a **WINSINTF_RESULTS_NEW_T** structure (section 2.2.2.11), which contains the results of the command execution.

Return Values: A 32-bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that operation completed successfully. A nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000005 ERROR_ACCESS_DENIED	The caller does not have sufficient permissions.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.

Exceptions Thrown: No exceptions SHOULD be thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

Processing and Response Requirements:

- The behavior of this method is exactly same as that of **R_WinsStatus** except for the following:
  - There is no limit on the number of entries in the address-version map array.
  - This method SHOULD NOT be called with **Cmd_e** set to WINSINTF_E_ADDVERSMAP. If it is, the server returns an ERROR_WINS_INTERNAL error.
- Refer to **R_WinsStatus** and **WINSINTF_RESULTS_NEW_T** for the details of the behavior of this method.

3.1.4.21 R_WinsStatusWHdl (Opnum 20)

The **R_WinsStatusWHdl** method retrieves various configuration settings and the statistics of a WINS server.

```
DWORD R_WinsStatusWHdl(
  [in, ref] WINSIF_HANDLE ServerHdl,
  [in] WINSINTF_CMD_E Cmd_e,
  [in, out, ref] PWINSINTF_RESULTS_NEW_T pResults
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to. See [MSDN-Handles] for more information.

This value MUST be ignored by the WINS server on receipt.

Cmd_e: The command to be executed on the target WINS server.

pResults: A pointer to a structure of type **WINSINTF_RESULTS_NEW_T (section 2.2.2.11)** that contains the results of the command execution.

Return Values: A 32-bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that operation completed successfully. A nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

The behavior of this method is the same as that of the **R_WinsStatusNew** method (section 3.1.4.20).

3.1.4.22 R_WinsDoScavengingNew (Opnum 21)

The **R_WinsDoScavengingNew** method requests a specific scavenging operation on the target WINS server.

```
DWORD R_WinsDoScavengingNew(
  [in] handle_t ServerHdl,
  [in, ref] PWINSINTF_SCV_REQ_T pScvReq
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to determine which WINS server the call is directed to.

pScvReq: A pointer to a WINSINTF_SCV_REQ_T structure (section 2.2.2.12) that defines the type of scavenging operation.

Return Values: A 32-bit unsigned integer that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that the operation completed successfully. A nonzero return value is a Win32 error code, as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000005 ERROR_ACCESS_DENIED	The caller doesn't have sufficient permissions.

Exceptions Thrown: No exceptions SHOULD be thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

Processing and Response Requirements:

When **R_WinsDoScanvengingNew** is called, the method returns immediately without waiting for scavenging to start. The server just queues a request for the scavenging operation; the internal

state and configuration of the WINS server and the value of the *fForce* member in the **WINSINTF_SCV_REQ_T** structure (section 2.2.2.12) determine whether the scavenging occurs. Hence, callers to **R_WinsDoScanvengingNew** SHOULD NOT treat a return code of ERROR_SUCCESS as indicating a successful scavenging operation. Instead, callers SHOULD rely on WINS event logs to determine whether or not the scavenging operation succeeded.

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsDoScavengingNew**:

- Callers to **R_WinsDoScavengingNew** SHOULD have control level access. If an RPC client with a lower access level calls **R_WinsDoScavengingNew**, the server SHOULD return ERROR_ACCESS_DENIED.
- The WINS server queues a request on the target WINS server for the scavenging operation, and the method returns immediately with ERROR_SUCCESS as the status code.

3.1.5 Timer Events

No protocol timer events are required other than those in the underlying RPC protocol.

3.1.6 Other Local Events

No local events are maintained other than those in the underlying RPC protocol.

3.2 winsi2 Server Details

The methods supported by the **winsi2** interface are specified in **Message Processing Events and Sequencing Rules (section 3.2.4)**.

3.2.1 Abstract Data Model

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

A NetBIOS name server (NBNS) needs to maintain the following data structures:

Name record: A data structure that contains a name and the associated attributes.

Name records collection: A collection of all name records that are either registered by this NBNS server or obtained by replication.

Global version counter: A 64-bit unsigned integer that tracks the version number that is given to the next record to be updated.

Server configuration: Parameters maintained in persistent storage include the following:

- Refresh interval
- Extinction interval
- Extinction timeout
- Verify interval

- Process priority class
- Number of worker threads

3.2.2 Timers

No timers are required beyond those used internally by RPC to implement resiliency to network outages, as specified in [MS-RPCE] section 3.2.3.2.1.

3.2.3 Initialization

A WINS **winsi2** remote protocol server MUST be initialized by registering the RPC interface and listening on the dynamically allocated port assigned by RPC, as specified in section 2.1. The client MUST connect to a well-known RPC port on the WINS server to determine the endpoint of **winsi2**. Before any client connection, the WINS server MUST wait for WINS winsi2 to register with RPC before any clients can establish a connection.

3.2.4 Message Processing Events and Sequencing Rules

The **winsi2** interface provides methods that remotely configure, manage, and monitor a WINS server.

Methods in RPC Opnum Order

Method	Description
R_WinsTombstoneDbRecs	Tombstones a specified range of records belonging to a particular owner. Opnum: 0
R_WinsCheckAccess	Checks the granted level of access for the RPC caller. Opnum: 1

3.2.4.1 R_WinsTombstoneDbRecs (Opnum 0)

The **R_WinsTombstoneDbRecs** method tombstones records whose version numbers fall within a range of version numbers and are owned by a server with a specified address.

```
DWORD R_WinsTombstoneDbRecs(
  [in] handle_t ServerHdl,
  [in, ref] PWINSINTF_ADD_T pWinsAdd,
  [in] WINSINTF_VERS_NO_T MinVersNo,
  [in] WINSINTF_VERS_NO_T MaxVersNo
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to resolve which WINS server the call is directed to.

pWinsAdd: A pointer to the address of the owner WINS server whose records are to be tombstoned. This value MUST NOT be NULL.

MinVersNo: The lower bound on the range of version numbers that identifies the range of records to be tombstoned.

MaxVersNo: The upper bound on the range of version numbers that identifies the range of records to be tombstoned.

Return Values: A 32 bit unsigned integer value that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that the operation completed successfully. Any other return value is a Win32 error code as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.
0x00000005 ERROR_ACCESS_DENIED	The caller does not have sufficient permissions.
0x00000FA0 ERROR_WINS_INTERNAL	An error occurred while processing the RPC call.

Exceptions Thrown: No exceptions SHOULD be thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

Processing and Response Requirements:

The following requirements and recommendations apply to a WINS server that processes a call to **R_WinsTombstoneDbRecs**:

- The **R_WinsTombstoneDbRecs** caller SHOULD have control-level access. If an RPC client with a lower access level calls this method, the server SHOULD return ERROR_ACCESS_DENIED.
- If the specified owner WINS server address is not found in the owner-version map table, the server SHOULD return ERROR_WINS_INTERNAL.
- If any error occurs during the retrieval or updating of database records, the server SHOULD return ERROR_WINS_INTERNAL.
- The server changes the state of the matching records to tombstoned. It also updates the version number and the ownership of these records so that the version number and record ownership are replicated on the partner WINS servers when replication takes place.
- The time stamp of the matching record is set to a string with the following format:

```
current time + tombstone timeout configured on the target WINS server
```

- If both *MinVersNo* and *MaxVersNo* are zero, all records matching the given owner address are tombstoned.

3.2.4.2 R_ WinsCheckAccess (Opnum 1)

The **R_ WinsCheckAccess** method retrieves the level of access the client is granted.<12>

```
DWORD R_WinsCheckAccess(
  [in] handle_t ServerHdl,
  [out] DWORD* Access
);
```

ServerHdl: An RPC binding over IP address/HostName to the WINS server. RPC uses this binding internally to resolve which WINS server the call is directed to.

Access: Pointer to the access level value. This value MUST not be NULL. The following values are possible as output.

Name	Value
No access	0
Control level access	1
Query level access	2

Return Values: A 32-bit unsigned integer value that indicates the return status. A return value of ERROR_SUCCESS (0x00000000) indicates that the operation completed successfully. Any other return value is a Win32 error code as specified in [MS-ERREF]. The following Win32 error codes can be returned:

Return value/code	Description
0x00000000 ERROR_SUCCESS	The call was successful.

Exceptions Thrown: No exceptions SHOULD be thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

Processing and Response Requirements:

Clients with any access level can call this method.

3.2.5 Timer Events

No protocol timer events are required other than those in the underlying RPC protocol.

3.2.6 Other Local Events

No local events are maintained other than those in the underlying RPC protocol.

4 Protocol Examples

4.1 Inserting a Record into a WINS Database

The following example illustrates the use of the RPC methods defined in this specification to insert a record into the database of a WINS server. If the WINS database on the specified server does not have a record with name "WINS-TEST-00001", then the client calls the RPC method **R_WinsRecordAction (section 3.1.4.1)** with the following parameters:

- *ServerHdl* set to the endpoint of the WINS server on which the **R_WinsRecordAction** method is executed.
- *ppRecAction* pointing to a structure of type **WINSINTF_RECORD_ACTION_T (section 2.2.2.3)**, with members set as follows:
  - **Cmd_e** to WINSINTF_E_INSERT.
  - **pName** to point to the string "WINS-TEST-00001" followed by a NetBIOS suffix of 0x00.
  - **NameLen** to 16.
  - **TypOfRec_e** to 3, indicating a multihomed record.
  - **NoOfAdds** to 2.
  - **pAdd** to point to two IP addresses: 192.168.1.1 and 192.168.1.2.
  - **NodeTyp** to 1, indicating a p-node.
  - **fStatic** to 0, indicating a dynamic record.

4.2 Releasing a Record from a WINS Database

The following example illustrates the use of the RPC methods defined in this specification to release a record from the database of a WINS server. If the WINS database on the specified server has a unique record with name "WINS-TEST-00001" mapped to the IP address 192.168.1.1, then the client calls the RPC method **R_WinsRecordAction (section 3.1.4.1)** with the following parameters:

- *ServerHdl* set to the endpoint of the WINS server on which the **R_WinsRecordAction** method is executed.
- *ppRecAction* pointing to a structure of type **WINSINTF_RECORD_ACTION_T (section 2.2.2.3)**, with members set as follows:
  - **Cmd_e** to WINSINTF_E_RELEASE.
  - **pName** to point to the string "WINS-TEST-00001" followed by a NetBIOS suffix of 0x00.
  - **NameLen** to 16.
  - **TypOfRec_e** to 0.
  - **NoOfAdds** to 1.
  - **Add** to the IP address 192.168.1.1.

4.3 Deleting a Record from a WINS Database

The following example illustrates the use of the RPC methods defined in this specification to delete a record from the database of a WINS server. If the WINS database on the specified server has a multihomed record with name "WINS-TEST-00001", the client calls the RPC method **R_WinsRecordAction (section 3.1.4.1)** with the following parameters:

- *ServerHdl* set to the endpoint of the WINS server on which the **R_WinsRecordAction** method is executed.
- *ppRecAction* pointing to a structure of type **WINSINTF_RECORD_ACTION_T (section 2.2.2.3)**, with members set as follows:
  - **Cmd_e** to WINSINTF_E_DELETE.
  - **pName** to point to the string "WINS-TEST-00001" followed by a NetBIOS suffix of 0x00.
  - **NameLen** to 16.
  - **State_e** to 3 (DELETED).

4.4 Modifying a Record from a WINS Database

The following example illustrates the use of the RPC methods defined in this specification to modify a WINS server database record. If the WINS database on the specified server has a multihomed dynamic record with the name "WINS-TEST-00001" and the node type set to p-node, the client calls the RPC method **R_WinsRecordAction (section 3.1.4.1)** with the following parameters:

- *ServerHdl* set to the endpoint of the WINS server on which the **R_WinsRecordAction** method is executed.
- *ppRecAction* pointing to a structure of type **WINSINTF_RECORD_ACTION_T (section 2.2.2.3)**, with members set as follows:
  - **Cmd_e** to WINSINTF_E_MODIFY.
  - **pName** to point to the string "WINS-TEST-00001" followed by a NetBIOS suffix of 0x00.
  - **NameLen** to 16.
  - **TypOfRec_e** to 3, indicating a multihomed record.
  - **NodeTyp** to 3.
  - **State_e** to 1.
  - **fStatic** to zero.

After executing the call to **R_WinsRecordAction**, the node type and the state of the existing record are modified to h-node and RELEASED, respectively.

4.5 Querying a Record from a WINS Database

The following example illustrates the use of the RPC methods defined in this specification to query a record from the database of a WINS server. This example assumes that the WINS database contained by the specified server has an active multihomed dynamic record named "WINS-TEST-00001" mapped to two IP addresses: 192.168.1.1 and 192.168.1.2. IP address 192.168.1.1 has

node type p, and its time stamp set to 0x61. IP address 192.168.1.2 has a version ID type and a time stamp set to 0x101E. The client calls the RPC method **R_WinsRecordAction (section 3.1.4.1)** with the following parameters:

- *ServerHdl* set to the endpoint of the WINS server on which the **R_WinsRecordAction** method is executed.
- *ppRecAction* pointing to a structure of type **WINSINTF_RECORD_ACTION_T (section 2.2.2.3)**, with members set as follows:
  - **Cmd_e** to WINSINTF_E_QUERY.
  - **pName** to point to the string "WINS-TEST-00001" followed by a NetBIOS suffix of 0x00.
  - **NameLen** to 16.
- All other members are used for output, which the server assigns as follows:
  - **TypOfRec_e** contains 3.
  - **NoOfAdds** contains 2.
  - **pAdd** points to two IP addresses: 192.168.1.1 and 192.168.1.2.
  - **VersNo** contains 0x61.
  - **NodeTyp** contains 1.
  - **OwnerId** contains the owner IP address of the matching record.
  - **State_e** contains zero (ACTIVE).
  - **fStatic** contains zero.
  - **TimeStamp** contains 0x101E.

4.6 Retrieving All of the Records of a WINS Database

This example illustrates the use of the RPC methods defined in this specification to retrieve all the records from the database of a WINS server.

- The client calls the **R_WinsGetDbRecsByName** method repeatedly with the following parameters.
- Set *pWinsAdd* to NULL, *Location* to zero, *pName* to NULL, *NameLen* to zero, *fStaticOnly*, to 4 and *NoOfRecsDesired* to the desired number of records. As noted in the description of **R_WinsGetDbRecsByName**, the server resets the *NoOfRecsDesired* parameter to 5,000 if the parameter's value is greater than 5,000.
- Check how many **R_WinsGetDbRecsByName** has returned by looking at the value in the *NoOfRecs* field. If this value is less than the *NoOfRecsDesired* value, the retrieval is complete. Otherwise, if the number of returned values is the same as the value of *NoOfRecsDesired*, call to the **R_WinsGetDbRecsByName** with the following parameter settings:
- Set *pWinsAdd* to NULL, *Location* to 0, *pName* to the name of the last record retrieved in the previous iteration, *NameLen* to the length of *pName*, *fStaticOnly* to 4, and *NoOfRecsDesired* to the desired number of records.

- Repeated this procedure until the value of *NoOfRecs* is less than the value of *NoOfRecsDesired*.

4.7 Deleting All the Records of an Owner from a Particular WINS Server

This example illustrates the use of the RPC methods defined in this specification to delete all the records of an owner from the target WINS server.

The client calls the RPC method **R_WinsDelDbRecs** with the following parameters:

- The endpoint of the WINS server on which **R_WinsDelDbRecs** is executed (or from which the records are deleted as *ServerHdl*.
- Set *MinVersNo*, *MaxVersNo* and *pAdd->Type* to zero 0, *pAdd->Len* to 4, and *pAdd->IPAdd* to the IP address of the WINS server whose records are to be deleted.
- The successful completion of the **R_WinsDelDbRecs** call deletes all the records for an owner from the target WINS server database.

4.8 Deleting All the Records from a Particular WINS Server

This example illustrates the use of the RPC methods defined in this specification to delete all records from the target WINS server.

The client calls the RPC method **R_WinsStatusNew** with the following parameters:

- The endpoint of the WINS server on which the RPC method is executed, or from which the records are deleted, as *ServerHdl*.
- Set *Cmd_e* to *WINSINTF_E_CONFIG_ALL_MAPS*.
- The output of the call to **R_WinsStatusNew**, *pResults*, contains the list of owner addresses in the database of the target WINS server. For each owner address in *pResults->pAddVersMaps*, call the RPC method **R_WinsDelDbRecs (section 3.1.4.9)** by setting the parameters as follows:
  - Set *MinVersNo*, *MaxVersNo*, and *pAdd->Type* to 0.
  - Set *pAdd->Len* to 4.
  - Set *pAdd->IPAdd* to pResults->pAddVersMaps[*i*]->Add, where **i** denotes the *i*th iteration.

4.9 Triggering a Pull Replication Between Two WINS Servers

This example illustrates the use of the RPC methods defined in this specification to trigger a pull replication from one WINS server to another.

The client calls the RPC method **R_WinsTrigger** with the following parameters:

- Set the value of *ServerHdl* to the endpoint of the WINS server on which the pull replication is queued.
- Set *TrigType_e* and *pAdd->Type* to 0, *pAdd->Len* to 4 and *pAdd->IPAdd* to the IP address of the WINS server that serves as the partner for the pull replication.
- A return value of ERROR_SUCCESS means that the pull request has been queued successfully.

4.10 Backing Up a WINS Server Database

To back up the WINS server database, the client calls the RPC method **R_WinsBackup** with the following parameters:

- Set the value of *ServerHdl* to the endpoint of the WINS server on which the backup is performed.
- Set *pBackupPath* to the path on the server where the database is backed up.
- A return value of ERROR_SUCCESS indicates that the backup has been successful.

5 Security

5.1 Security Considerations for Implementers

RAIW allows any user to establish a connection to the RPC server. The protocol uses the underlying RPC protocol to retrieve the identity of the method caller as specified in [MS-RPCE]. Clients create an authenticated RPC connection, and servers use this identity to perform specific access checks.

WINS server data and WINS server operations specified by this implementation are protected by access checks based on the identity of the RPC client.

Servers that implement this specification do not allow anonymous RPC connections and protect WINS access to all data and operations with access control checks based on client identity.

Clients or servers that implement this specification do not use RPC over named pipes because it is vulnerable to man-in-the-middle attacks. RPC over TCP/IP is used instead.

Servers that implement this protocol require clients to request RPC_C_AUTHN_WINNT, and servers enforce this requirement in order to protect the privacy of the communication with clients.

5.2 Index of Security Parameters

Security parameter	Section
RPC_C_AUTHN_WINNT	Section 2.1.1

6 Appendix A: Full IDL

6.1 winsif Interface

For ease of implementation, the full stand-alone Interface Definition Language (IDL) file for the winsif interface (section 3.1) is provided. Some of the data types and structures used by this protocol are defined in other documents. In order for this IDL to stand alone, the types and structures from [MS-DTYP] are imported.

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

#define WINSINTF_MAX_NO_RPL_PNRS 25

typedef PVOID LPVOID;
typedef LARGE_INTEGER WINSINTF_VERS_NO_T;

typedef struct _WINSINTF_ADD_T {
   BYTE Type;
   DWORD Len;
   DWORD IPAdd;
} WINSINTF_ADD_T, *PWINSINTF_ADD_T;

typedef enum _WINSINTF_PRIORITY_CLASS_E {
   WINSINTF_E_NORMAL = 0,
   WINSINTF_E_HIGH
} WINSINTF_PRIORITY_CLASS_E, *PWINSINTF_PRIORITY_CLASS_E;

typedef enum  _WINSINTF_ACT_E {
   WINSINTF_E_INSERT = 0,
   WINSINTF_E_DELETE,
   WINSINTF_E_RELEASE,
   WINSINTF_E_MODIFY,
   WINSINTF_E_QUERY
} WINSINTF_ACT_E, *PWINSINTF_ACT_E;

typedef enum _WINSINTF_TRIG_TYPE_E {   WINSINTF_E_PULL = 0,
   WINSINTF_E_PUSH,
   WINSINTF_E_PUSH_PROP
} WINSINTF_TRIG_TYPE_E, *PWINSINTF_TRIG_TYPE_E;

typedef struct _WINSINTF_RECORD_ACTION_T {
   WINSINTF_ACT_E Cmd_e;
   [size_is(NameLen + 1)] LPBYTE pName;
   DWORD NameLen;
   DWORD TypOfRec_e;
   DWORD NoOfAdds;
   [unique, size_is(NoOfAdds)] PWINSINTF_ADD_T pAdd;
   WINSINTF_ADD_T Add;
   LARGE_INTEGER VersNo;
   BYTE NodeTyp;
   DWORD OwnerId;
   DWORD State_e;
   DWORD fStatic;
   DWORD_PTR TimeStamp;
} WINSINTF_RECORD_ACTION_T, *PWINSINTF_RECORD_ACTION_T;

typedef struct _WINSINTF_RPL_COUNTERS_T {
```

```
    WINSINTF_ADD_T Add;
    DWORD NoOfRpls;
    DWORD NoOfCommFails;
} WINSINTF_RPL_COUNTERS_T, *PWINSINTF_RPL_COUNTERS_T;

typedef struct _WINSINTF_STAT_T {
    struct {
        DWORD NoOfUniqueReg;
        DWORD NoOfGroupReg;
        DWORD NoOfQueries;
        DWORD NoOfSuccQueries;
        DWORD NoOfFailQueries;
        DWORD NoOfUniqueRef;
        DWORD NoOfGroupRef;
        DWORD NoOfRel;
        DWORD NoOfSuccRel;
        DWORD NoOfFailRel;
        DWORD NoOfUniqueCnf;
        DWORD NoOfGroupCnf;
    } Counters;
    struct {
        SYSTEMTIME   WINSStartTime;
        SYSTEMTIME LastPScvTime;
        SYSTEMTIME LastATScvTime;
        SYSTEMTIME LastTombScvTime;
        SYSTEMTIME LastVerifyScvTime;
        SYSTEMTIME LastPRplTime;
        SYSTEMTIME LastATRplTime;
        SYSTEMTIME LastNTRplTime;
        SYSTEMTIME LastACTRplTime;
        SYSTEMTIME LastInitDbTime;
        SYSTEMTIME CounterResetTime;
    } TimeStamps;
    DWORD NoOfPnrs;
    [unique, size_is(NoOfPnrs)] PWINSINTF_RPL_COUNTERS_T pRplPnrs;
} WINSINTF_STAT_T, *PWINSINTF_STAT_T;

typedef struct _WINSINTF_ADD_VERS_MAP_T {
    WINSINTF_ADD_T Add;
    LARGE_INTEGER VersNo;
} WINSINTF_ADD_VERS_MAP_T, *PWINSINTF_ADD_VERS_MAP_T;

typedef struct _WINSINTF_RESULTS_T {
    DWORD NoOfOwners;
    WINSINTF_ADD_VERS_MAP_T AddVersMaps[WINSINTF_MAX_NO_RPL_PNRS];
    LARGE_INTEGER MyMaxVersNo;
    DWORD RefreshInterval;
    DWORD TombstoneInterval;
    DWORD TombstoneTimeout;
    DWORD VerifyInterval;
    DWORD   WINSPriorityClass;
    DWORD NoOfWorkerThds;
    WINSINTF_STAT_T   WINSStat;
} WINSINTF_RESULTS_T, *PWINSINTF_RESULTS_T;

typedef struct _WINSINTF_RESULTS_NEW_T {
    DWORD NoOfOwners;
    [unique, size_is(NoOfOwners)]
        PWINSINTF_ADD_VERS_MAP_T pAddVersMaps;
```

```
    LARGE_INTEGER MyMaxVersNo;
    DWORD RefreshInterval;
    DWORD TombstoneInterval;
    DWORD TombstoneTimeout;
    DWORD VerifyInterval;
    DWORD   WINSPriorityClass;
    DWORD NoOfWorkerThds;
    WINSINTF_STAT_T   WINSStat;
} WINSINTF_RESULTS_NEW_T, *PWINSINTF_RESULTS_NEW_T;

typedef enum _WINSINTF_CMD_E {
    WINSINTF_E_ADDVERSMAP = 0,
    WINSINTF_E_CONFIG,
    WINSINTF_E_STAT, WINSINTF_E_CONFIG_ALL_MAPS
} WINSINTF_CMD_E, *PWINSINTF_CMD_E;

typedef struct _WINSINTF_RECS_T {
    DWORD BuffSize;
    [unique,size_is(NoOfRecs)] PWINSINTF_RECORD_ACTION_T pRow;
    DWORD NoOfRecs;
    DWORD TotalNoOfRecs;
} WINSINTF_RECS_T, *PWINSINTF_RECS_T;

typedef struct _WINSINTF_PULL_RANGE_INFO_T {
    LPVOID pPnr;
    WINSINTF_ADD_T OwnAdd;
    WINSINTF_VERS_NO_T MinVersNo;
    WINSINTF_VERS_NO_T MaxVersNo;
} WINSINTF_PULL_RANGE_INFO_T, *PWINSINTF_PULL_RANGE_INFO_T;

typedef struct _WINSINTF_BROWSER_INFO_T {
    DWORD dwNameLen;
    [string] LPBYTE pName;
} WINSINTF_BROWSER_INFO_T, *PWINSINTF_BROWSER_INFO_T;

typedef struct _WINSINTF_BROWSER_NAMES_T {
    DWORD EntriesRead;
    [unique, size_is(EntriesRead)]  PWINSINTF_BROWSER_INFO_T pInfo;
} WINSINTF_BROWSER_NAMES_T, *PWINSINTF_BROWSER_NAMES_T;

typedef enum _WINSINTF_SCV_OPC_E {
    WINSINTF_E_SCV_GENERAL,
    WINSINTF_E_SCV_VERIFY
} WINSINTF_SCV_OPC_E, *PWINSINTF_SCV_OPC_E;

typedef struct _WINSINTF_SCV_REQ_T {
    WINSINTF_SCV_OPC_E Opcode_e;
    DWORD Age;
    DWORD fForce;
} WINSINTF_SCV_REQ_T, *PWINSINTF_SCV_REQ_T;

typedef struct _WINSINTF_BIND_DATA_T {
    DWORD fTcpIp;
    [string] LPSTR pServerAdd;
    [string] LPSTR pPipeName;
 } WINSINTF_BIND_DATA_T, *PWINSINTF_BIND_DATA_T;

[
```

```
    uuid(45F52C28-7F9F-101A-B52B-08002B2EFABE),
    version(1.0),
    pointer_default(unique)
]

interface winsif {

#define MIDL_PASS

typedef [handle] PWINSINTF_BIND_DATA_T WINSIF_HANDLE;
typedef handle_t WINSIF2_HANDLE;

#define DECLARE_WINS_HANDLE(_hdl ) [in] WINSIF2_HANDLE _hdl,
#define DECLARE_WINS_HANDLE0(_hdl ) [in] WINSIF2_HANDLE _hdl

DWORD R_WinsRecordAction(
           DECLARE_WINS_HANDLE( ServerHdl )
    [in, out, ref] PWINSINTF_RECORD_ACTION_T *ppRecAction
);

DWORD R_WinsStatus(
           DECLARE_WINS_HANDLE( ServerHdl )
    [in]   WINSINTF_CMD_E          Cmd_e,
    [in, out, ref] PWINSINTF_RESULTS_T pResults
);

DWORD R_WinsTrigger(
           DECLARE_WINS_HANDLE( ServerHdl )
    [in, ref] PWINSINTF_ADD_T pWinsAdd,
    [in]   WINSINTF_TRIG_TYPE_E  TrigType_e
);

DWORD R_WinsDoStaticInit(
           DECLARE_WINS_HANDLE( ServerHdl )
    [in, unique, string] LPWSTR pDataFilePath,
    [in]   DWORD fDel
);

DWORD R_WinsDoScavenging(
           DECLARE_WINS_HANDLE0( ServerHdl )
);

DWORD R_WinsGetDbRecs(
           DECLARE_WINS_HANDLE( ServerHdl )
    [in, ref] PWINSINTF_ADD_T pWinsAdd,
    [in]   WINSINTF_VERS_NO_T MinVersNo,
    [in]   WINSINTF_VERS_NO_T MaxVersNo,
    [out]  PWINSINTF_RECS_T pRecs
);

DWORD R_WinsTerm(
    [in]   handle_t ServerHdl,
    [in]   short fAbruptTem
);

DWORD R_WinsBackup(
           DECLARE_WINS_HANDLE( ServerHdl )
    [in, string, ref] LPBYTE pBackupPath,
    [in]   short fIncremental
```

```
);

DWORD R_WinsDelDbRecs(
            DECLARE_WINS_HANDLE( ServerHdl )
    [in, ref] PWINSINTF_ADD_T pWinsAdd,
    [in]   WINSINTF_VERS_NO_T MinVersNo,
    [in]   WINSINTF_VERS_NO_T MaxVersNo
);

DWORD R_WinsPullRange(
            DECLARE_WINS_HANDLE( ServerHdl )
    [in, ref] PWINSINTF_ADD_T pWinsAdd,
    [in, ref] PWINSINTF_ADD_T pOwnerAdd,
    [in]   WINSINTF_VERS_NO_T MinVersNo,
    [in]   WINSINTF_VERS_NO_T MaxVersNo
);

DWORD R_WinsSetPriorityClass(
            DECLARE_WINS_HANDLE( ServerHdl )
    [in]   WINSINTF_PRIORITY_CLASS_E PrCls_e
);

DWORD R_WinsResetCounters(
            DECLARE_WINS_HANDLE0( ServerHdl )
);

DWORD R_WinsWorkerThdUpd(
            DECLARE_WINS_HANDLE( ServerHdl )
    [in]   DWORD NewNoOfNbtThds
);

DWORD R_WinsGetNameAndAdd(
            DECLARE_WINS_HANDLE( ServerHdl )
    [out, ref] PWINSINTF_ADD_T pWinsAdd,
    [out, string, size_is(80)] LPBYTE pUncName
);

DWORD R_WinsGetBrowserNames_Old(
            DECLARE_WINS_HANDLE( ServerHdl )
    [out] PWINSINTF_BROWSER_NAMES_T pNames
);

DWORD R_WinsDeleteWins(
            DECLARE_WINS_HANDLE( ServerHdl )
    [in, ref] PWINSINTF_ADD_T pWinsAdd
);

DWORD R_WinsSetFlags(
            DECLARE_WINS_HANDLE( ServerHdl )
    [in]   DWORD fFlags
);

DWORD R_WinsGetBrowserNames(
    [in, ref] WINSIF_HANDLE ServerHdl,
    [out] PWINSINTF_BROWSER_NAMES_T pNames
);

DWORD R_WinsGetDbRecsByName(
            DECLARE_WINS_HANDLE( ServerHdl )
```

```
    [in, unique] PWINSINTF_ADD_T pWinsAdd,
    [in]   DWORD Location,
    [in, unique, size_is(NameLen + 1)] LPBYTE pName,
    [in]   DWORD NameLen,
    [in]   DWORD NoOfRecsDesired,
    [in]   DWORD fOnlyStatic,
    [out]  PWINSINTF_RECS_T pRecs
);

DWORD R_WinsStatusNew(
           DECLARE_WINS_HANDLE( ServerHdl )
    [in]   WINSINTF_CMD_E Cmd_e,
    [out]  PWINSINTF_RESULTS_NEW_T pResults
);

DWORD R_WinsStatusWHdl(
    [in, ref] WINSIF_HANDLE ServerHdl,
    [in] WINSINTF_CMD_E Cmd_e,
    [in, out, ref] PWINSINTF_RESULTS_NEW_T pResults
);

DWORD R_WinsDoScavengingNew(
           DECLARE_WINS_HANDLE( ServerHdl )
    [in, ref] PWINSINTF_SCV_REQ_T pScvReq
);
}
```

6.2 winsi2 Interface

For ease of implementation, the full stand-alone Interface Definition Language (IDL) file for the winsi2 interface (section 3.1) is provided. Some of the data types and structures used by this protocol are defined in other documents. In order for this IDL to stand alone, the types and structures from the winsif interface (section 6.1) IDL are imported.

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

[
    uuid(811109bf-a4e1-11d1-ab54-00a0c91e9b45),
    version(1.0),
    pointer_default(unique)
]

interface winsi2 {

#define MIDL_PASS

typedef handle_t WINSIF2_HANDLE;

DWORD
R_WinsTombstoneDbRecs(
    [in]      WINSIF2_HANDLE ServerHdl,
    [in, ref] PWINSINTF_ADD_T pWinsAdd,
    [in]      WINSINTF_VERS_NO_T MinVersNo,
    [in]      WINSINTF_VERS_NO_T MaxVersNo
);
```

```
DWORD
R_WinsCheckAccess(
    [in]      WINSIF2_HANDLE ServerHdl,
    [out]     DWORD *Access
);
}
```

7 Appendix B: Product Behavior

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

- Microsoft Windows NT® 4.0 operating system
- Microsoft Windows® 2000 operating system
- Windows® XP operating system
- Windows Server® 2003 operating system
- Windows Server® 2008 operating system
- Windows Server® 2008 R2 operating system

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

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

<1> Section 2.1.1: Windows NT 4.0: The Remote Administrative Interface: WINS protocol does not define query-level access; the only access level available is control-level access. Clients that invoke RPC have control-level access.

<2> Section 2.2.2.7: Windows 2000, Windows Server 2003, Windows Server 2008, and Windows Server 2008 R2: If the value of the **WINSPriorityClass** member is other than NORMAL_PRIORITY_CLASS or HIGH_PRIORITY_CLASS, the system assumes the latter.

<3> Section 3.1.4.1: Windows NT 4.0: The Remote Administrative Interface: WINS Protocol uses implicit binding, in which the RPC run-time library maintains the handle internally. No RPC methods except **R_WinsTombStoneDbRecs**, **R_WinsTerm**, and **R_WinsGetBrowserNames** take *ServerHdl* as a parameter.

<4> Section 3.1.4.1: Windows NT 4.0: The RPC method caller is required to have control-level access regardless of the action used.

<5> Section 3.1.4.1: Windows 2000, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2: A maximum of 25 IP address mappings are allowed for a multihomed or special group Name Record.

<6> Section 3.1.4.2: Windows NT 4.0: The **R_WinStatus** caller is required to have control-level access regardless of the command used.

<7> Section 3.1.4.6: Windows NT 4.0: The RPC method caller is required to have control-level access regardless of the command used.

<8> Section 3.1.4.10: The execution of **R_WinsPullRange** by a client with sufficient access permissions can cause the WINS service on the target WINS server to restart.

<9> Section 3.1.4.14: Windows NT 4.0: The **R_WinsGetNameAndAdd** caller is required to have control-level access regardless of the command used.

<10> Section 3.1.4.14: Windows NT 4.0: The **R_WinsGetNameAndAdd** caller does not need access permissions to call this method.

<11> Section 3.1.4.19: Windows NT 4.0: The RPC method caller is required to have control-level access regardless of the command used.

<12> Section 3.2.4.2: Windows NT 4.0: This RPC method is not supported.

8 Change Tracking

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

9 Index

R

S

T

V

W