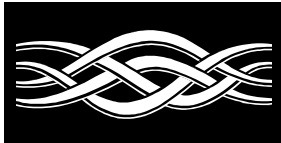




Microsoft®

Microsoft®
Windows NT® Server

System Operating System



White Paper

**Microsoft® Windows NT® Server 4.0
Windows Internet Naming Service (WINS)**

Architecture and Capacity Planning

© 1996 Microsoft Corporation. All rights reserved.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Microsoft, Windows, Windows NT, and the BackOffice logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Other product and company names mentioned herein may be the trademarks of their respective companies.

Microsoft Corporation ■ One Microsoft Way ■ Redmond, WA 98052-6399 ■ USA

1096 Part no. 098-67425

Microsoft® **Windows NT® Server**

WINS

The Windows Internet Naming Service Architecture and Capacity Planning

This paper describes the Microsoft® Windows Internet Naming Service (WINS) technology. WINS provides dynamic NetBIOS name to IP address registration and resolution. This paper presents a functional description of the WINS server. Name registration and resolution processes are described as well as replication between WINS servers. The WINS database schema is documented. Finally, the paper discusses capacity planning and designing a WINS infrastructure.

CONTENTS

INTRODUCTION.....	1
NetBIOS Names	1
NetBIOS-Based Networks	1
LMHOSTS File	2
Enhancements to LMHOSTS file	2
Using centralized LMHOSTS files	3
Limitations of LMHOSTS file	3
WINDOWS INTERNET NAME SERVICE (WINS).....	5
WINS Server	5
WINS Client	5
Benefits of Using WINS	5
WINS/DNS Integration	6
FUNCTIONAL DESCRIPTION.....	7
Name Registration, Refresh, and Release	7
Name Registration	7
Name Refresh	8
Name Release	8
Conflicts detected during registration	9
Name Registration	10
Name Refresh	10
Name release	11
Name reregistration with the same WINS server	11
Name Reregistration with a different WINS server	12
Extinction process	13
Name Query	13
Querying a WINS server directly	14
Proxy Servers	14
Querying with Proxy Server	16
Secondary WINS Servers	16
TIMERS	17
Renewal Interval	17
Extinction Interval	17
Extinction Time-out	17
Verify Interval	17
Server Clocks	18
Note:	18
Static Entries	18
Migrate On	19
Groups and Multihomed Names	19
Normal groups	19
Special groups	20
WINS Manager	20

LMHOSTS	20
Multihomed names	21
Multihomed name initialization through the LMHOSTS file	21
REPLICATION	23
Overview of the Replication Process	24
Detailed Replication Example	25
Pulling WINS Database Entries	28
What gets replicated	30
Conflicts Detected During Replication	30
Conflict between unique entries	31
Conflict between two replicas with the same or different IP addresses	31
Conflict between owned entry and replica with same IP address	31
Conflict between owned entry and replica with the different IP addresses	31
Conflict between a unique entry and a group entry	31
Conflict between two Special Group entries	31
Conflict involving a Multihomed record	31
Scavenging	32
NEW IN NT 4.0	34
Burst Mode	34
Administering WINS Through a Firewall	34
Consistency Check by WINS	35
WINSCHK	35
WINS DATABASE	38
IP Address - Owner ID Mapping Table	38
IP address	38
Owner Identifier	38
Name to IP Address Mapping Table	38
Name	38
Addresses	38
Flags field	39
Owner id	39
Version ID	39
Time Stamp	40
Winsdmp	40
DESIGNING A WINS INFRASTRUCTURE	41
Practical Consideration for Design	41
Design criteria	41
Convergence time	41
Fault tolerance	42
Duplicate replication traffic	42
Server size	43

Database size	43
Server Performance	44
Some configuration examples	44
Network Traffic	46
Typical network traffic	46
WINS server and WINS client traffic	46
Replication and verify traffic	47
Typical machine behavior	47
Turning a machine off overnight	47
Roving user	47
Stationary machines	48
Joining two WINS systems	48
Estimates of the total amount of traffic	48
SUMMARY	49
For more information	49
APPENDIX A: TERMINOLOGY AND NETBIOS NAMES	50
Terminology	50
NetBIOS Names	50
Unique Names	50
Group Names	51
APPENDIX B: NETBT (NETBIOS OVER TCP) CONFIGURATION	
PARAMETERS	52
Introduction	52
Standard Parameters Configurable from the Registry Editor	52
Optional Parameters Configurable from the Registry Editor	53
Parameters Configurable from the Network Control Panel Applet	55
Non-Configurable Parameters	56
APPENDIX C: NAME REGISTRATION TRACE	58
Part 1 Name Registration Request, WACK and Registration rResponse	59
Part 2 Name Refresh Request and Response	65

INTRODUCTION

NetBIOS Names

In order to understand the architecture of WINS, it is first necessary to understand the history behind it: that is, NetBIOS. NetBIOS started as a high-level programming language interface for PC-DOS applications to IBM PC-Network broadband LANs. Microsoft® used this NetBIOS interface for designing its networking components. NetBIOS names identify resources and are 16 characters in length. The NetBIOS name space is flat, meaning that names may only be used once within a network. These names are registered dynamically when computers boot, services start, or users log on. NetBIOS names can be registered as unique or as group names. Unique names have one address associated with a name.¹ Group names have more than one address mapped to a name.

Microsoft networking components, such as Windows NT® Workstation and Windows NT Server services, allow the first 15 characters of a NetBIOS name to be specified by the user or administrator, but reserve the 16th character (the suffix) of the NetBIOS name (00-FF hex) to indicate a resource type. (See Appendix A for a list of Microsoft reserved suffixes.) Some third-party applications use different suffixes.

NetBIOS-Based Networks

NetBIOS is a session-level interface that is used by applications to communicate over NetBIOS compatible transports. It is responsible for establishing logical names on the network, establishing sessions between two logical names on the network, and supporting reliable data transfer between computers that have established a session. NetBIOS has been around for more than a decade. Those protocols implemented under Microsoft networking components, including TCP/IP, have a NetBIOS interface or a mapping layer (NetBIOS over TCP/IP is called NetBT). This layer is provided to allow non-native NetBIOS components to fit into a NetBIOS environment. NetBIOS-based communications use NetBIOS names to uniquely identify resources and other nodes on the network.

Name resolution in a NetBIOS network has traditionally been broadcast-based. A Name Registration Request is broadcast and heard by all B- and M-nodes² on the local network. If, after the request is made, no objections are received, the application assumes that it has permission to use the name and issues a Name Overwrite Demand. If the name is already in use, a Negative Name Registration Response datagram is received from the node holding the name. In this case, the requesting application does not have permission to use the name.

There are several disadvantages to a broadcast-based name resolution system. First, nodes may interoperate with one another within a broadcast area but cannot interoperate across routers in a routed network. Second, such a system generates high-broadcast traffic, using significant network bandwidth. Third, every node within the broadcast area must examine each broadcast datagram. This consumes resources on every node.

Broadcast-based name resolution works fine within a small LAN environment. As

¹ Per interface on a multi-homed computer.

² See Appendix A: Terminology and NetBIOS Names

the LAN grows and eventually merges into a WAN, however, this method is not effective. Large LANs will begin to experience bandwidth problems and once routers are introduced, the system becomes inoperable.

LMHOSTS File

The LMHOSTS file was introduced to assist with remote NetBIOS name resolution. The LMHOSTS file is a static file that maps NetBIOS names to IP addresses. This is similar to the hosts file in functionality; the only difference is that the hosts file is used for mapping host names (hierarchical names) to IP addresses. A NetBIOS name to IP address mapping in the LHHOSTS file allows that NetBIOS name to be resolved for a node that cannot respond to name query broadcasts.

Computers in a Microsoft-based network can resolve NetBIOS names in several different ways. If one method of resolution fails, they try the next method in a fixed order. In a broadcast-based network, the node first checks its remote name cache (the name will be in the cache if it has been used recently or preloaded from LMHOSTS) before broadcasting a name query. As a last resort it uses the LMHOSTS file to obtain the name to IP address mapping (for example, to resolve a name across a router in a broadcast-based network).

Enhancements to LMHOSTS file The LMHOSTS file introduced initially had limitations and could not be used to support functions such as logon validation across subnets or designing a domain spanning subnets.

The above limitations were removed by introducing a few keywords in the Windows NT 3.1 LMHOSTS file. The keywords and their functionality are described below.

Keyword	Description
#PRE	Added after an entry to cause that entry to be preloaded into the name cache. The entry is locked in the name cache and does not time out. By default, entries are not preloaded into the name cache but are parsed only after WINS and name query broadcasts fail to resolve a name. The #PRE keyword must be appended for entries that also appear in #INCLUDE statements; otherwise, the entry in #INCLUDE is ignored.
#DOM:<domain>	Added after an entry to associate that entry with the domain specified by <domain>. This keyword affects how the Browser and Logon services behave in routed TCP/IP environments. To pre-load a #DOM entry, you must also add the #PRE keyword to the line.
#INCLUDE <filename>	Forces the system to seek the specified <filename> and parse it as if it were local. Specifying a Uniform Naming Convention (UNC) <filename> allows you to use a centralized LMHOSTS file on a server. If the server is located outside of the local broadcast area, you must add a mapping for the server before its entry in the #INCLUDE section and also append the #PRE keyword to ensure that it preloaded.

Keyword	Description
#BEGIN_ALTERNATE	Used to group multiple #INCLUDE statements. Any single successful #INCLUDE causes the group to succeed.
#END_ALTERNATE	Used to mark the end of an #INCLUDE statement grouping.

Using centralized LMHOSTS files The keywords described in the previous section can be used to allow name resolution across small- to medium-sized, routed networks. With Microsoft TCP/IP, an LMHOSTS file can include other LMHOSTS files from local and remote computers using the #INCLUDE statement. Network administrators can maintain one or more global LMHOSTS files for inclusion in local LMHOSTS files. These global LMHOSTS files may be distributed to multiple servers for reliable access by using the existing Windows NT Replicator service.

#BEGIN_ALTERNATE and #END_ALTERNATE keywords can be used to provide a redundant list of servers maintaining copies of the same LMHOSTS file. This is known as a *block inclusion*, which allows multiple servers to be searched for a valid copy of a specific file. The following example shows the use of these keywords.

```

167.148.45.20 testdc #PRE #DOM:TEST #primary DC
167.148.45.21 testbdc #PRE #DOM:TEST #backup DC in domain

#INCLUDE c:\private\lmhosts #include a local lmhosts

#BEGIN_ALTERNATE
#INCLUDE \\testdc\public\lmhosts #source for global file
#INCLUDE \\testbdc\public\lmhosts #backup source
#END_ALTERNATE

```

In this example, the servers testdc and testbdc are located on subnets remote from the computer that owns the file. During name resolution, the Windows NT system first includes this private file, then gets the global LMHOSTS file from one of two locations: testdc or testbdc.³ The first available source for the global LMHOSTS satisfies the block inclusion and its entries are included. Any other servers in the list are not used.

Limitations of LMHOSTS file Despite the many uses of the LMHOSTS file, there are some limitations to its design. Its greatest limitation is that it is a static file. Because the LMHOSTS file is static, entries have to be updated if the name or the IP address of the computer changes. An IP address might change for several reasons. Physically moving the computer between subnets would require a new IP address, as would a remote user dialing-in via the Remote Access Server (RAS) in Windows NT Server. A new address might also be acquired from a DHCP Server after lease expiration.

³ All names of servers in the #INCLUDE statements must have their addresses preloaded using the #PRE keyword; otherwise, the #INCLUDE statement is ignored.

In addition to modifying the LMHOSTS file when such changes occur, these changes must be propagated to all the computers needing access to the resource whose name-to-IP address mapping has been modified. A centrally maintained LMHOSTS file can reduce the manual administration effort involved in propagating these mappings to the required computers. However, entering and changing the mappings is still a manual, labor intensive process. This is certainly preferable to keeping an LMHOSTS file on each computer and this solution can be feasible for a company with 10 to 20 servers. With more servers it becomes too complicated to maintain and is not a viable solution for a large enterprise.

This limitation of the LMHOSTS file has been exacerbated by the introduction of the Dynamic Host Configuration Protocol (DHCP). A DHCP server assigns IP addresses to nodes dynamically. It is nearly impossible to keep the LMHOSTS file updated when IP addresses are being assigned dynamically.

WINDOWS INTERNET NAME SERVICE (WINS)

WINS provides a distributed database for registering and querying dynamic NetBIOS names to IP address mapping in a routed network environment. It is the best choice for NetBIOS name resolution in such a routed network because it is designed to solve the problems that occur with name resolution in complex Internetworks.

The LMHOSTS file addressed only one disadvantage of broadcast based systems—it allowed resolution of names across routers. Since the system itself was still broadcast-based, the problems of broadcast traffic and load on local nodes were not solved. RFCs 1001/1002 address these problems. They define a protocol that allows name registration and resolution through unicast datagrams to NetBIOS Name Servers (NBNS). Because unicast datagrams are used, the system inherently works across routers. This eliminates the need for an LMHOSTS file, restoring the dynamic nature of NetBIOS name resolution. This, in turn, allows the system to work seamlessly with DHCP. For example, when dynamic addressing through DHCP results in new IP addresses for computers that move between subnets, the changes are automatically updated in the WINS database. Neither the user nor the network administrator needs to make manual accommodations for name resolution in such a case.

The WINS protocol is based on and is compatible with the protocols defined for NBNS in RFCs 1001/1002, so it is interoperable with other implementations of these RFCs. Another RFC-compliant implementation of the client can talk to the WINS server, and similarly, a Microsoft TCP/IP client can talk to other implementations of the NBNS server. However, because the WINS server-to-server replication protocol is not specified in the standard, the WINS server will not interoperate with other implementations of a NetBIOS Name Server. Data will not be replicated between the WINS server and the non-WINS NBNS. Therefore the WINS system as a whole will not converge and name resolution will not be guaranteed. WINS consists of two main components, the WINS server and WINS client.

WINS Server

- Handles name registration/release requests from WINS clients and registers/releases their names and IP addresses.
- Responds to name queries from WINS clients by returning the IP address of the name being queried (assuming the name is registered with the WINS server).
- Replicates the WINS database with other WINS servers.

WINS Client

- Registers/releases its name with the WINS server when it joins/leaves the network.
- Queries the WINS server for remote name resolution.

Benefits of Using WINS

- Dynamic database maintenance to support computer name registration and resolution.
- Centralized management of NetBIOS name database.

-
- Reduction of IP broadcast traffic in the Internetwork, while allowing the clients to locate remote systems easily across local or wide-area networks.
 - The ability for the clients (Windows NT 3.5 (or newer), Windows® for Workgroups 3.11, Windows® 95) on a Windows NT Server-based network to browse remote domains without a local domain controller being present on the other side of the router.
 - On a Windows NT network, the ability to browse transparently across routers (for domains that span multiple subnets). To allow browsing without WINS, the network administrator must ensure that the users' primary domain has Windows NT Server or Windows NT Workstation computers on both sides of the router to act as master browsers. These computers need correctly configured LMHOSTS files with entries for the domain controllers across the subnet.

WINS/DNS Integration

In Windows NT 4.0 Microsoft's implementation of DNS is tightly integrated with WINS. This allows non-WINS clients to resolve NetBIOS names by querying a DNS server. Administrators can now remove any static entries for Microsoft-based clients in legacy DNS server zone files in favor of the dynamic WINS/DNS integration. For example, if a non-Microsoft-based client wants to get to a Web page on an HTTP server that is DHCP/WINS enabled, the client can query the DNS server, the DNS server can query WINS and the name can be resolved and returned to the client. Previous to the WINS integration, there was no way to reliably resolve the name because of the dynamic IP addressing. Please see the *DNS and Microsoft Windows NT 4.0* white paper for details on the WINS/DNS integration.

FUNCTIONAL DESCRIPTION

The following functional description specifically discusses name registration and resolution within a WINS system. Broadcast-based systems may be mentioned from time to time for comparison but they will not be discussed in detail.

In a WINS system, all names are registered with a WINS server. The names are stored in a database on the WINS server which answers requests for name-to-IP address resolution based on the entries in this database. Redundancy and load balancing are maintained by allowing more than one WINS server in the WINS system. These servers replicate their database entries among one another periodically in order to maintain a consistent view of the name space.

Each name has an entry in the database. It is owned by the WINS server it registered with and is a replica on all other WINS servers. Each entry has a state associated with it—the entry may be in the *active*, *released*, or *extinct* (also known as *tombstone*) state.⁴ Entries are also assigned a version ID. This number is used in the replication process and will be discussed later.

The WINS system also allows the registration of *static* names. This enables the administrator to register names for servers running operating systems that are not capable of dynamic name registration. WINS distinguishes between dynamic and static entries. Static names are treated somewhat differently than dynamic names (this will be discussed later).

Name Registration, Refresh, and Release

Name Registration Name Registration is a request for the use of a name. The request may be for a unique (exclusive) or a group (shared) name. NetBIOS applications may register one or more names. Names are acquired dynamically through the registration procedure; each application contends with other applications in real time.

In order to request a name, the client node sends a Name Registration Request directly to the WINS server. The WINS server accepts or rejects the name registration by issuing a Positive or Negative Name Registration Response to the requesting node. The action taken by the WINS server depends on several factors; whether the name already exists in the database and in what state, whether it has the same or a different address, and whether the request is for a unique or group entry.

If the name does not exist in the database, it is accepted as a new registration. The name is entered with a new version ID, a Time Stamp of Current Time + Renewal Interval, and the WINS server's owner ID. A Positive Name Registration Response is sent. The Renewal Interval reflects the Time to Live (TTL) of the name. This is discussed in more detail in the sections on Name Refresh and Timers.

If the name is already entered in the database with the same IP address as that being requested, the action taken depends on state and ownership. If the state is active and the entry is owned by this WINS server, the Time Stamp will be updated and a Positive Name Registration Response will be sent. If the entry is in the released or tombstone state or if the entry is owned by another WINS server, the registration is treated as new. Time stamp, version ID, and ownership are all

⁴ Replica entries may only be *active* or *extinct*. They are never in the *released* state.

updated and a Positive Name Registration Response is sent.

In the case where the name exists in the WINS database but with a different IP address than that being requested, the WINS server must be careful to avoid duplicate names. If the database entry is in the *released* or *tombstone* state, the WINS server is free to assign that name. The request is treated as a new registration and a Positive Name Registration Response is sent. If, however, the entry is in the *active* state, the node holding the name must be challenged to see if it still exists on the network.⁵ The WINS server first sends a Wait for Acknowledgment (WACK) Response to the requesting node, specifying a time in the TTL field that the client should be prepared to wait for a response. The WINS server then issues a Name Query Request to the node registered in the database. If the node still exists, it sends a Positive Name Query Response back to the WINS server. The WINS server, in turn, sends a Negative Name Registration Response back to the requesting node, rejecting the name registration. If a Positive Name Query Response is received, the WINS server will send the Name Query Request two additional times at approximately a 500 ms interval. After three attempts with no response, a Positive Name Registration Response is returned to the requesting node and the name is registered in the database as a new registration.

Name Refresh Names held by WINS are given a Time to Live (TTL) or Renewal Interval during name registration. A name must be refreshed before this interval ends or it will be released. Names are refreshed by sending a Name Refresh Request to the WINS server.⁶ It is the responsibility of the client to refresh the name before the Renewal Interval expires. Windows NT clients refresh at 1/2 of the Renewal Interval. Other client stacks may refresh at different frequencies. The client node may increase the refresh rate if the WINS server does not respond to the refresh request. The WINS server treats a Name Refresh in the same way as a Name Registration.

Name Release NetBIOS names may be explicitly or silently released. Names are explicitly released when a node shuts down gracefully. A silent release typically occurs when an end node fails or is powered off. The silent release is noted at the WINS server when a name is not refreshed within the *Renewal Interval*.

When a name is released, the database entry is marked as *released* and Time Stamped with *Current Time + Extinction Interval*. This information is not propagated to partner WINS servers. If the release is explicit, the WINS server makes itself the owner of the record if it is not already.

In Windows NT 4.0 the *release* is handled differently if the Owner ID is different (the client is releasing to a different WINS server than it registered with). In this case the entry is marked as *extinct*. The Time Stamp is *Current Time + Extinction Interval + Extinction Time-out*. This is done to avoid windows of inconsistencies between a secondary and primary WINS server. Without doing this, the name would

⁵ If the database entry is a group the name registration is rejected. Groups are never challenged. Member addresses are challenged in sequence for multihomed nodes.

⁶ In RFC 1002, the opcode field of a Name Refresh Request is defined as 0x8. However, an example in the same RFC shows a Name Refresh Request with opcode 0x9. The Microsoft TCP/IP stacks send opcode 0x8 in a Name Refresh Request. Some stacks, such as those from Ungermann-Bass send opcode 0x9. Because of this conflict, WINS will recognize opcodes of either 0x8 or 0x9 as a Name Refresh Request.

stay released on one WINS server and active on the other for longer than desired periods because a released record would not be replicated again since it has already been replicated once. Changing the released record to the extinct state results in its replication and brings WINS databases into sync faster for this potentially troublesome case. For example, if a client's primary WINS server were unavailable when the client was shut down, the name release would be directed to the secondary WINS server. If the primary WINS server were available again when the client rebooted, the client would register and continue to refresh with the primary WINS server which still believes the client to be in the active state. The secondary WINS server database would keep the client in the released state.

Conflicts detected during registration When a client node registers or refreshes a name, that name may already exist in the WINS database. The action taken by the WINS server depends on what state the registered name is in. The name might be unique or a group name. It might be owned or replica, static or dynamic. It might be active, released, or extinct (tombstone). The IP address might be the same or different.

Two cases are always the same and very simple. Normal group entries and Static entries are never overwritten.⁷ The WINS server always returns a Negative Name Registration Response to a request for a name that is entered in the database as a group or static name. Special groups get additional members during a Special group registration. In this case if the record in conflict is in the released or tombstone state, the record registration will be treated as new.

This leaves us with unique dynamic entries.

If the IP addresses are the same, the WINS server will return a Positive Name Registration Response and take the following action:

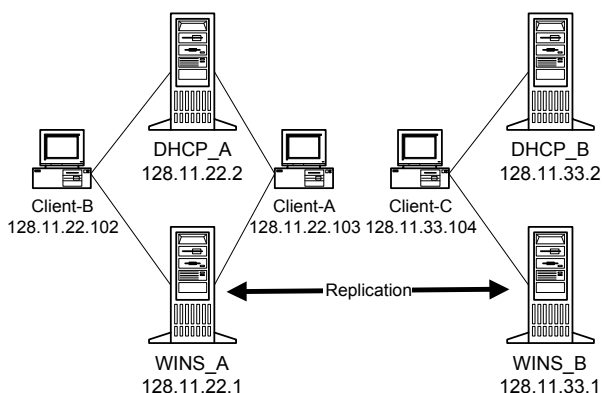
State	Action
Owned active	Time stamp
Replica active	Time stamp, take ownership, new version ID
Released	Time stamp, make active, new version ID
Owned tombstone	Time stamp, make active, new version ID
Replica tombstone	Time stamp, take ownership, make active, new version ID

If the IP addresses are different and the database record is released or tombstoned, the name registration will be treated as new. A Positive Name Registration Response will be sent and the entry updated to reflect the new time, ownership, new version ID, and active state.

If the database entry is in the active state with a different IP address, the WINS server must determine if the name with the old IP address still exists. A Name Query Request is sent to the old IP address. If the old address responds with a Positive Name Query Response, the WINS server, in turn, rejects the new

⁷ The Migrate On switch in WINS Administrator allows static entries (only unique/mh) to be overwritten after the challenge has revealed them to be obsolete by dynamic entries. This switch is specifically for the migration of static entries to dynamic and is not turned on in normal operation.

registration with a Negative Name Registration Response. If the old address does not respond to the Name Query, the new registration is accepted.



Name Registration In the previous diagram, let's assume that Client_B is a DHCP client. We will start our example on 1/18/96 at about 2:10 PM. The Renewal Interval is set at the default, which is four days. When we power on Client_B, the IP address, subnet mask, default gateway, primary (WINS_A), and secondary (WINS_B) WINS servers are received from the DHCP server. Client_B discovers the MAC address of the primary WINS server with an Address Resolution Protocol (ARP) request. A Name Registration Request is then sent by UDP datagram directly to WINS_A. Assume that, at this time, there is no entry for Client_B in the WINS_A database. WINS_A sends a Positive Name Registration Response to Client_B and a record for Client_B is entered in the WINS database. The Time Stamp is Current Time + Renewal Interval and the version ID is the value of the registration counter on WINS_A. The registration counter on WINS_A is then incremented by one. This entry will be replicated to WINS_B when the next Replication Interval timer expires (or when the update count, if specified when configuring a partner as a push partner for this WINS, is reached), if it is still in the active state at that time.⁸

Name	Address	Flags	Owner	Version ID	Time Stamp
ClientB	128.11.22.102	unique,active,h-node,dynamic	WINS_A	4B3	1/22/96 2:12:56 PM

In the Positive Name Registration Response, WINS_A sends a Time to Live (TTL). This is the *Renewal Interval* set by the administrator for WINS_A.

Name Refresh At 1/2 Renewal Interval (Windows NT client), Client_B sends a Name Refresh Request to WINS_A. WINS_A returns a Positive Name Registration Response and updates the database record with a new Time Stamp.

Name	Address	Flags	Owner	Version ID	Time Stamp
ClientB	128.11.22.102	unique,active,h-node,dynamic	WINS_A	4B3	1/22/96 5:24:56 PM

The WINS server treats a Name Refresh Request exactly the same as a Name

⁸ If Client_B released before replication and did not reregister before the extinction interval expired, the first replication would be in the extinct state.

Registration Request. If Client_B does not send a Name Refresh Request to WINS_A within the Renewal Interval, WINS_A will put Client_B in the released state.

Name Release The user shuts down his machine for the evening at 5:35 PM on 1/18/96. When Client_B is shut down gracefully, it issues a Name Release Request to WINS_A. WINS_A responds with a Positive Name Release Response. The database record is updated by placing Client_B in the released state and updating the Time Stamp. The Time Stamp in this case is Current Time + Extinction Interval. The Extinction Interval in this case is four days. The version ID is not changed at this time because a record in the released state is never replicated.

Name	Address	Flags	Owner	Version ID	Time Stamp
ClientB	128.11.22.102	unique, released, h-node, dynamic	WINS_A	4B3	1/22/96 5:35:36 PM

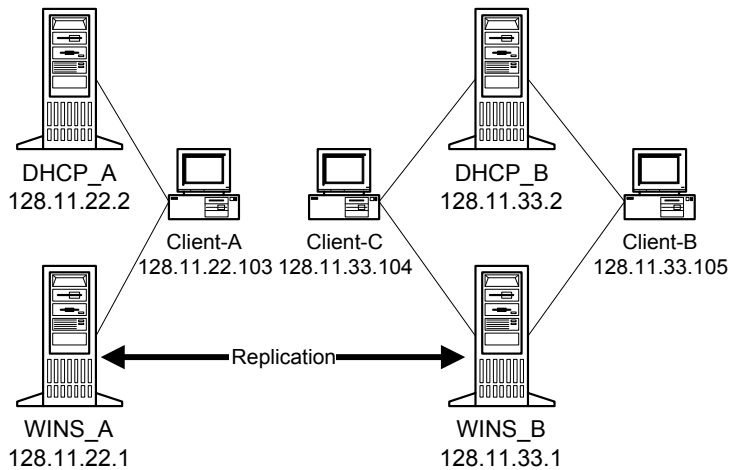
If Client_B is not reregistered before the Extinction Interval expires, it will become extinct (or a tombstone)

Name Reregistration with the same WINS server When the user reboots the next morning, 1/19/96, at 8:15 AM, Client_B will again send a Name Registration Request to WINS_A. This time, an entry for Client_B does exist in the WINS database. Because the entry is in the released state, WINS_A can issue a Positive Name Registration Response without a challenge (challenge will be discussed later). In the database, the entry is placed in the active state, the time is stamped with Current Time + Renewal Interval, and the version ID is given the value of the registration counter. Because of the new version ID, this entry will be replicated to other WINS_B when the next Replication Interval timer expires (or update count is reached).⁹

Name	Address	Flags	Owner	Version ID	Time Stamp
ClientB	128.11.22.102	unique, active, h-node, dynamic	WINS_A	4EF	1/23/96 8:15:46 PM

⁹ It would be preferable not to replicate the reregistration of a name with the same IP address. However, because entries in the released state are never replicated, it would be possible for an active entry never to be replicated if we didn't do this. If a name were registered and then released before replication, it would not be replicated. If this name were then reregistered after the database had been replicated, the existing version ID would be less than the last replicated version ID. We must bump the version ID in this case to force the replication of this active entry.

Name Reregistration with a different WINS server



In this example, let's say that Client_B is a laptop computer. At 12:30 PM on 1/19/96 the user shuts down Client_B (putting Client_B in the released state on WINS_A) and then goes to another building where Client_B is connected to a different subnet (at about 1:15 PM). When Client_B boots on this subnet, it receives WINS_B as its primary WINS server. Client_B sends a Name Registration Request to WINS_B. WINS_B already has an entry for Client_B that had been replicated from WINS_A.¹⁰

Name	Address	Flags	Owner	Version ID	Time Stamp
ClientB	128.11.22.102	unique,active,h-node,dynamic	WINS_A	4B3	2/12/96 8:20:56 PM

The existing entry shows WINS_A as the owner WINS server and has a different IP address than the current request.¹¹ In this case, WINS_B must determine if the name registered in its database still exists on the network. WINS_B sends a Wait for Acknowledgment (WACK) Response to the (new) Client_B with an estimated time necessary to complete the request. It then challenges the (old) Client_B by sending it a Name Query Request. If WINS_B were to receive a Positive Name Query Response from the (old) Client_B, it would send a Negative Name Registration Response to the (new) Client_B, denying the Name Registration. However, in this case, the (old) Client_B is no longer on the network (because it is a laptop that was moved). Therefore, WINS_B receives no response to its Name Query Request and may issue a Positive Name Registration Response to the (new) Client_B. The database entry is modified to indicate the new WINS owner and new IP address. It is Time Stamped with Current Time + Renewal Interval and the

¹⁰ If the entry from WINS_A had not yet replicated, WINS_B would issue a Positive Name Request Response immediately and a Name Conflict would be discovered and resolved at replication time.

¹¹ The Time Stamp is so far in the future because active replicas are time stamped with Current Time + Verify Interval. This is discussed under replication.

version ID is set to the registration counter from WINS_B.

Name	Address	Flags	Owner	Version ID	Time Stamp
ClientB	128.11.33.105	unique,active,h-node,dynamic	WINS_B	26C	1/23/96 1:15:21 PM

Extinction process To reduce network traffic, we do not want to replicate entries whenever a node is shut down. However, we also don't want the WINS database cluttered with entries for nodes that have been permanently removed. When Client_B is shut down (at 6:45 PM on 1/19/96), it is placed in the released state as discussed above and Time Stamped with Current Time + Extinction Interval.

Name	Address	Flags	Owner	Version ID	Time Stamp
ClientB	128.11.33.105	unique,released,h-node,dynamic	WINS_B	26C	1/23/96 6:45:54 PM

During this period it remains in the released state and is not replicated. If, after the Extinction Interval, the name still has not been reregistered, the entry is placed in the extinct (or tombstone) state at scavenge time and the time is stamped with Current Time + Extinction Time-out. In this case, we will assume that the Extinction Time-out has been set to 24 hr. Scavenging happened that day at 7:05 PM.

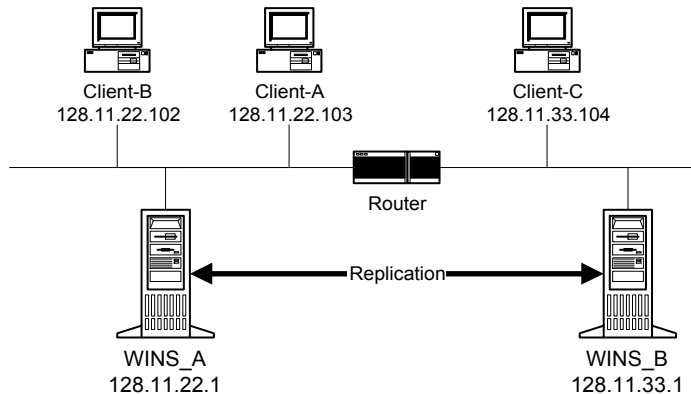
Name	Address	Flags	Owner	Version ID	Time Stamp
ClientB	128.11.33.105	unique,tombstone,h-node,dynamic	WINS_B	462	1/24/96 7:05:54 PM

At this point the version ID is also updated and the entry is replicated. The replica is also Time Stamped at the receiving WINS with Current Time + Extinction Time-out. If at the end of Extinction Time-out, the name is still not registered, it is removed from the WINS databases at scavenge time.

Name Query

Name Query transactions are initiated by end nodes to obtain the IP addresses of a NetBIOS name. The WINS server replies to queries with a list of IP addresses for each owner of the name (more than one address only if it is a Special Group or a multihomed name). The WINS server sends as many answers to the query as will fit into a UDP packet. Because Special Groups are limited to 25 names, all information will always fit into the UDP packet.

A Windows NT 3.5 computer only queried the secondary WINS server for NetBIOS name resolution when it did not receive a response from the primary WINS server. If a NetBIOS name was not found in the primary WINS database, Windows NT 3.5 did not search the secondary WINS server. The search order was changed in Windows NT 3.51. Now the secondary WINS server is queried when the Primary does not respond or when the requested name is not found in the primary WINS server database. This new search order also applies to Windows 95 and Windows for Workgroups 3.11 (using TCP/IP-32 for Windows for Workgroups version 3.11b and the updated redirector files).



Querying a WINS server directly In the previous diagram, when Client_B wishes to communicate with Client_C, the name Client_C must be resolved to the IP address 128.11.33.104. Client_B first looks in its local name cache for the name-to-IP address mapping. If Client_B has used the name recently, it will still be available in the cache. If the remote name cache does not contain the mapping, Client_B sends a Name Query Request to WINS_A. If WINS_A finds the name in its database it sends a Positive Name Query Response to Client_B with the IP address of Client_C. If the name is not found in the database, WINS_A sends a Negative Name Query Response to Client_B. There are several reasons why the name Client_C might not be in the database on WINS_A.

- Client_C has not yet been replicated from WINS_B to WINS_A.
The name cannot be resolved until replication takes place.
- Client_C is a b-node (broadcast only) and does not register with WINS.
The name could only be resolved if it were entered in a LMHOSTS file.¹²
- Client_C is a DNS node and does not have NetBIOS installed.
The name could be resolved if Client_B were configured to do DNS server lookups.

After receiving the Negative Name Query Response, Client_B's actions depend on which TCP/IP stack it is running. If it is running Microsoft's TCP/IP stack, it will query its secondary WINS server. If the secondary WINS server also issues a Negative Name Query Response Client_B will then, assuming it is configured as an h-node client (see Appendix A), broadcast a Name Query Request. The request will fail in this case because Client_C is on a different network and the router will not forward the broadcast. Next Client_B will do a LMHOSTS and finally a DNS lookup to resolve the name.¹³

Proxy Servers

RFC 1001 recommends that b-nodes (see Appendix A) not be used in a routed network. However, in practice, b-nodes are sometimes used in routed networks. In some cases, there are reasons why these b-nodes cannot be removed or updated

¹² The name could have been entered statically in the WINS database. In this case, the original query to WINS_A would have resolved the name.

¹³ The ability to do LMHOSTS and DNS lookups is configurable.

immediately. For this reason, Microsoft introduced proxy servers.

Proxy servers are nodes that listen for b-node name service functions (registration, release, query) and respond for those names that are not on the local network. Proxy servers communicate with the WINS server with directed datagrams to retrieve the information necessary to respond to these broadcasts.

Broadcast name registrations are not stored by the proxy, nor are they registered in the WINS database. They are simply checked against the WINS database (by sending a Name Query Request) to ensure that the names do not conflict with other names in the database. If the name exists in the WINS database, the proxy may send a Negative Name Registration Response to the b-node attempting to register the name. The action of sending a negative response is configurable. By default it is disabled. See `EnableProxyRegCheck` in Appendix A.

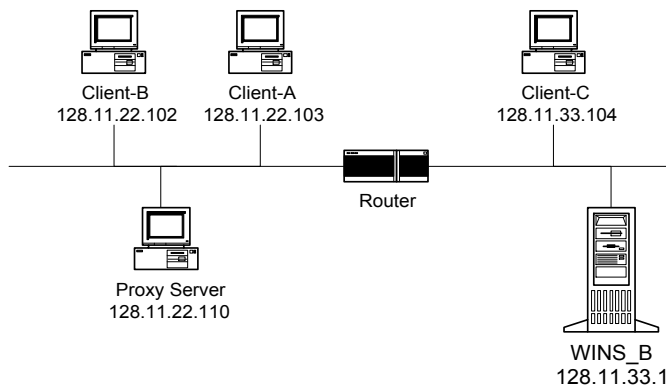
In response to a Name Release Request, the proxy simply deletes the name from its remote name cache.

When the proxy receives a name query it checks its remote name table. The proxy must be able to differentiate name queries for local names on the subnet and remote names elsewhere in the network. It compares the address of names it resolves to its own address using the subnet mask, and if the two match, then the proxy does not respond to the name query. This guarantees that the proxy will not respond to queries for names local to the subnet.

If the name is found in the remote name cache, a response is sent to the client. If the name cannot be found, the WINS server is queried and the name is entered into the remote name table in a "resolving" state. Should another query come in for the same name before the WINS server has responded, the proxy will not query the WINS server again. When the response is returned from the WINS server, the remote table entry is updated with the correct address and the state is changed to "resolved." When the next name query comes in for that name, a response is sent to the client. There is no provision in NetBIOS for a name server to "deliver" a name resolution to a client. A name is always resolved in response to a query, therefore the b-nodes used with the proxy should have configurable retry time-outs and number of retries in order to still be retrying the name query after the WINS server has responded to the proxy server.

In order to reduce duplicate traffic, it is recommended that only one or two proxy servers be active on a subnet.

Querying with Proxy Server



In the example shown above, we have a small broadcast-based LAN consisting of Client_A and Client_B connected to a larger network through a router. A NetBIOS application on Client_B wishes to communicate with Client_C. Normally, this would not be possible. However, by using a Windows NT Workstation-based client as a proxy server on the same LAN as Client_B, Client_B and Client_C can communicate.

Client_B broadcasts a Name Query Request to obtain the IP address of Client_C. Client_C does not receive the request because the router does not pass the broadcast. The proxy server sees a Name Query Request broadcast for a node on a different subnet and sends a Name Query Request directed datagram to WINS_B. WINS_B returns a Positive Name Query Response containing the IP address for Client_C to the proxy server where it is cached for future queries.

Secondary WINS Servers

Client nodes are configured with a primary and secondary WINS server. If the primary WINS server cannot be reached for any function (registration, refresh, release, query), the client will request that function from its secondary WINS server. The client will try periodically to switch back to its primary WINS server.

TIMERS

There are four configurable timer values. **Microsoft has chosen the defaults with care and in general they should not be modified.** They are designed to keep network traffic and the load on WINS servers at a minimum. They represent the best tradeoff, considering the various configurations in which WINS servers may be deployed, between these goals and keeping the window, during which the databases remain out of synch, small. Considerations such as long weekends, avoidance of unnecessary replication traffic, ability to handle large number of clients quickly under worst case scenarios (for example all PCs come up during a short time interval), as well as tradeoffs between quick clutter removal and retaining entries to ensure replication have been factored into the determination of these intervals.

Renewal Interval

This is also known as the Name Refresh Time-out or the Time to Live (TTL). When a name is registered with the WINS server, the database entry is Time Stamped with Current Time + Renewal Interval. A client must refresh its name with the WINS server within this interval or the name will be released. When a name is released (either by timing out or the by the explicit release of the name by the client), no action is taken other than changing the entry to the released state and Time Stamping the entry with Current Time + Extinction Interval. This change is not replicated to other WINS servers. When a name is in the released state and a new registration comes in with a different address, the name can be immediately given to the new client without challenge because it is known that the old client is no longer using the name. The default Renewal Interval is four days (six days in Windows NT 4.0).

Extinction Interval

This is also known as the Name Age Time-out or the Tombstone Interval. This is the interval at which released names are changed to the extinct (or tombstone) state. At this time the entry is Time Stamped with Current Time + Extinction Time-out and the version ID of the entry is updated causing this information to be propagated to all WINS servers at the next replication time. When extinct (or tombstone) entries are created, they are Time Stamped with Current Time + Extinction Time-out at the pulling WINS server. The default Extinction Interval is based on renewal and replication times. This is typically four days in Windows NT 3.51 and six days in Windows NT 4.0.

Extinction Time-out

This is also known as Tombstone Time-out. Extinct (or tombstone) records that are older than Extinction Time-out are removed from the database. The default Extinction Time-out is based on a number of factors. This default is also typically four days in Windows NT 3.51 and six days in Windows NT 4.0.

Verify Interval

Replication should ensure that the databases stay synchronized. However, under

certain abnormal conditions, names no longer in use could remain in the database. We refer to this as a *database clutter*. For example, if a tombstone record is removed before being replicated, the active state of the record in the replica databases would never change. This could happen if the replication partner was not reachable during the extinction time-out period.

When an active entry is replicated, it is Time Stamped with Current Time + Verify Interval on the pulling WINS server. If, at scavenge (see section on Scavenging for a definition) time, records are found that are older than the Verify Interval, a query is sent to the WINS server that owns that name asking if the version ID is still valid. A negative response (invalid record) causes the record to be removed. A positive response (valid record) causes the Time Stamp to be updated. If the WINS server cannot be contacted, the entries are left until the next Verify Interval (or until the administrator triggers scavenging). No records are removed if the owning WINS server cannot be contacted. The default Verify Interval is twenty-four days.

Server Clocks

The replication and scavenging algorithms rely on a reasonable consistent system clock. Setting the system clock forward or backward will, of course, affect these algorithms. However, because the Time Stamps are always entered locally, the WINS servers do not need to be time synchronized. The time only has to be consistent on each individual server.

Note: The above timer parameters are not strictly static. The values configured by the administrator may be altered by WINS in some situations. The algorithms by which WINS alters these parameters may be changed at any time by the WINS developers and therefore can not be definitively documented. As an example only, the algorithm for determining these values was at one time (version 3.5) as follows:

- Minimum Renewal Interval
 - defined to be 40 min. The default is four days.
- Extinction Interval
 - $\min(\max((2 * \text{max. Replication Interval}), \text{Renewal Interval}), \text{four days})$
 - It is also not allowed to go below $2 * \text{minimum Renewal Interval}$.
- Extinction Timeout
 - $\max(\max(\max(\min. \text{Renewal Interval}, \text{one day}), 4 * \text{max Replication Interval}), \text{Renewal Interval})$
- Verify Interval is
 - $\max(\max(24 \text{ days}, 3 * \text{Extinction Interval determined earlier}), \max(\text{<specified value>, 0}))$

Static Entries

Many nodes on a network, such as a server running the UNIX operating system, are not capable of registering a name with the WINS server. These names might be resolved from a LMHOSTS file or by querying a DNS server, but a better solution would be to enter the name to IP address mapping statically in the WINS server. This accomplishes two things. First, it allows nodes to resolve the name with a

query to the WINS server without having to resort to secondary resolution methods. This results in faster name resolution. Second, it prevents the WINS server from allowing another node to dynamically register the name.

Static entries may be entered interactively or by importing an LMHOSTS file. They are never released and they are never overwritten by dynamic entries.¹⁴

Migrate On The sole purpose of *Migrate On* is to ease the process of making a B-node an H-, P-, or M-node (WINS client). If set, the static nature of unique/mh names is compromised by WINS in that these records are now treated as pseudo-static. Pseudo-static means that a dynamic name registration will be allowed to overwrite a unique/mh static entry if the static's address(es) are different and prove to be wrong (the challenge mechanism is used to confirm this). If there were no concept of pseudo-static, the registrations would fail and the machine registering the names would not be able to come up properly.

The static nature of 1C groups is not compromised through *Migrate On* because these entries do not prevent any node from coming up. A static 1C entry can only be changed by an administrator. Dynamic name registrations that conflict with a static 1C name will return with success (to prevent name in conflict problem at the client) but the address in the registration will not be added to the member list in the WINS database.

Groups and Multihomed Names

In addition to unique entries, the WINS server allows groups and multihomed names to be registered. The WINS server recognizes two types of groups; normal groups and special groups.

Normal groups A normal group name does not actually have an address associated with it. It is assumed to be valid on any subnet. The same group can be registered at more than one WINS server. The Time Stamp on the group entry reflects the time for the last registration or refresh received for that group. The whole group has a single time associated with it which indicates the last time a name registration or refresh for the name was received from a node on any subnet. On a name query for the group, WINS will return the limited broadcast address. The WINS client will then issue a subnet broadcast to resolve the name.

As the group names replicate from one WINS server to another, the name is simply created on servers that do not already have it since there is no address to propagate with the name. When a group is not refreshed, it is released and eventually becomes a tombstone. These states, however, have a slightly different meaning for groups than they do for unique entries. The WINS server continues to answer name queries for released and tombstone groups. Unique name registrations that clash with these are returned a negative response. These states for group entries may be thought of as *pseudo states* (*pseudo-released* and *pseudo-tombstone*). At Extinction Interval, the version ID is incremented which causes the state information to be replicated to other WINS servers.

¹⁴ It is possible to allow unique/mh static entries to be overwritten by dynamic entries. This is done by setting the Migrate On switch in the WINS Administration interface. However, this switch is not on in normal operation.

Special groups Special groups are also known as Internet groups. When a name registration is received for a special group, the actual address rather than the limited broadcast address, will be stored in the group. A Time Stamp and an Owner ID will be stored with each address entry in the group. This reflects the last registration/refresh received for that entry by the WINS server is designated by the Owner ID. When a name query is received for such a group, the IP addresses that haven't timed out will be returned. These groups, like normal groups, get replicated. Only active members of the group are replicated. In Windows NT Server 3.5x, there is only one special group. This is the <domain_name>[1Ch]. This name is registered for use by the domain controllers within a domain and may contain up to 25 IP addresses. When queried, the IP address of the Primary Domain Controller (assuming the domain [0x1B] name was registered) will be followed by up to 24 IP addresses of Backup Domain Controllers will be returned. If this is not a static name and it has fewer than 25 entries, names registering dynamically will be added to it. If the group already has 25 entries, the WINS server will remove a replicated entry so that the new member can be added. If all the entries are owned by the WINS server (no replicated entries), the oldest member will be removed so that the new member may be added. WINS does not distinguish between local and remote domain controllers. It stores the members in the order of most recently refreshed to least recently refreshed. The [1Ch] domain name is used by the backup domain controllers to locate the PDC and is used when pass-through authentication is needed to validate a logon request. A statically mapped Internet group name registers itself as this type.

Unfortunately, there is no way to view the IP addresses contained in an Internet group with the WINS Manager, unless the Internet group is statically mapped in the WINS database (The group can be viewed with the winscl.exe program in the Resource Kit). The IP address that is displayed for this registration is the IP address of the last WINS Client to register/renew the Internet group name. When a dynamic registration for an Internet group is received, the WINS client's address and Time Stamp are stored in the Internet group.

In Windows NT 4.0, user defined special groups other than <domain_name>[1Ch] are allowed. An administrator can specify special group names of his/her own choice. There are two ways of doing this. One is through the WINS manager tool. The other is through the LMHOSTS file.

WINS Manager In the "Add Mappings" window, enter a name and check the "Internet Group" radio button.

LMHOSTS Specify the name just as you would a domain name except that the keyword is SG instead of DOM. An example is given below

```
128.11.80.182 nodea #SG:mygroup
```

The above results in a special group - mygroup<0x20>, being created with one member 128.11.80.182. Three unique records nodea<0x3>, nodea<0x20>, nodea<0x0> are also created.

The above is similar to what happens when you have the following:

```
128.11.80.182 nodea #DOM:mygroup
```

In this case, a domain mygroup<0x1C) is created with 157.55.80.182 as a

member. three unique entries nodea<0x3>, nodea<0x20>, nodea<0x0> are also created.

Specifying the name of a special group without specifying an address, can be done by giving the name of the group prefixed by #SG: #SG:mygroup. You can not do this for domains. That is, you can not have #DOM:mygroup to specify mygroup<0x1C> as a domain.

Addresses entered via the LMHOSTS file or via WINSADMN become permanent addresses in the special group and can be removed only through WINS Manager. In the case of special groups, excluding domains, members can get added as a result of dynamic group registrations. A dynamic member will however never replace a permanent member added via LMHOSTS or WINS manager.

Multihomed names A multihomed node can register one or more addresses by sending them in a name registration packet with the opcode set to a Microsoft defined value. This opcode is not prescribed by RFC 1001/1002 since these RFCs do not discuss multihomed clients. The opcode is one of the unused values in the 4 bit opcode field.

A multihomed name in the database of a WINS can have one or more addresses in it. When a refresh is received for a multihomed name and the address in the refresh packet is found in the multihomed name's record, the member address's Time Stamp is updated. The Time Stamp of the record is also updated so that it continues to reflect the most recent refresh/registration received for one or more members of the name. When the member address refreshed is found to be owned by another WINS, its ownership is changed to that of the local WINS and the version stamp of the record is updated.

When a registration is received for a multihomed name or when a refresh comes in with an address not in the list of addresses of the multihomed name in the database, each address in the list other than the address received in the registration/refresh is challenged in sequence until a positive response is received or until all addresses have been exhausted. If a positive response is received, the list of one or more addresses sent in the response is examined to determine if the list contains the address in the registration/refresh. If it does, a Positive Name Registration Response is returned to the client and the record in the database is appropriately updated—those addresses not there are inserted, those there have their Time Stamp and Owner ID updated. If however the list of addresses in the response is not a superset of the list found for the multihomed name, a Negative Name Registration response is returned to the client and the record in the database is left untouched. If all addresses in the list are exhausted without getting a positive response (or no response), the record in the database is overwritten with the new mapping and a positive response is returned to the client.

Multihomed name initialization through the LMHOSTS file In Windows NT 4.0 it is possible to initialize multihomed names in the LMHOSTS file. The format of giving the name is exactly the same as for a unique name except that the keywords #MH must be put after the name as is shown below.

```
128.11.80.182    nodea_ptm #MH
```

```
128.11.80.185    nodea_ptm #MH
```

The above example creates MH records nodea_ptm<0x0>, nodea_ptm<0x20>, nodea_ptm<0x3> with two addresses.

```
128.11.80.181    nodea_ptm #MH #DOM:REDMOND
```

```
128.11.80.185    nodea_ptm #MH #DOM:REDMOND
```

The example above creates MH records nodea_ptm<0x0>, nodea_ptm<0x20>, nodea_ptm<0x3> with two addresses and creates a domain REDMOND<0x1c> with the two addresses as members.

```
128.11.80.181    nodea_ptm #MH #SG:MYGROUP
```

```
128.11.80.185    nodea_ptm #MH #SG:MYGROUP
```

The above example creates MH records nodea_ptm<0x0>, nodea_ptm<0x20>, nodea_ptm<0x3> with two addresses. Creates a special group MYGROUP with the two addresses as members.

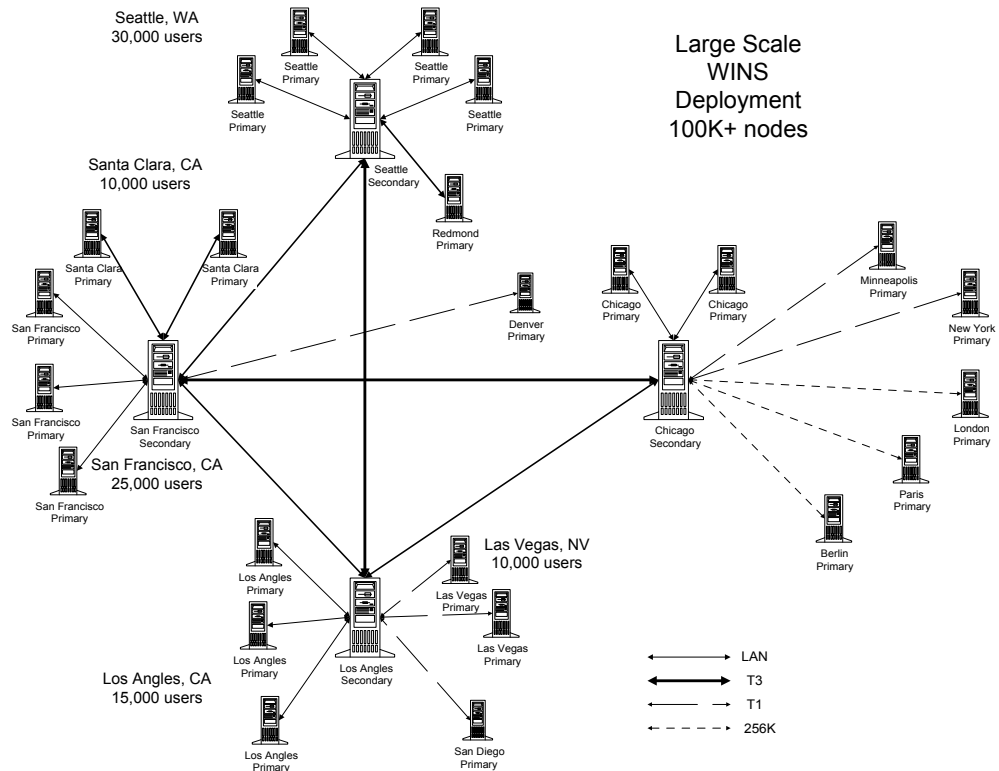
REPLICATION

Multiple WINS servers increase availability and balance the load among servers. The WINS system must maintain a stable name space. If a node has registered a name-to-address mapping with one WINS server, that mapping must be available reliably from any WINS server. This is accomplished through replication of the WINS databases among WINS servers. Replication is carried out among partners, rather than each server replicating to all other servers. Each WINS server must be configured with at least one other WINS server as a replication partner. This ensures that a name registered with one WINS server is eventually replicated to all other WINS servers. Replication of registered names to all WINS servers is necessary to allow resolution of names registered to different servers. All name-to-address mapping changes will converge within the Convergence Time of the entire WINS system. A name release will not propagate as fast as a name registration. This is because it is common for names to be released and then reused with the same mapping as PCs are rebooted or turned off for the evening or the weekend. Replicating each of these releases would unnecessarily increase the network load of replication. If a client node crashes or is simply powered off, the registered name is not cleanly released by a NetBIOS Name Release datagram. Therefore, the presence of a name-to-address record in the WINS database does not necessarily mean that the node is running. It only means that at some reasonable time in the past, that node claimed the IP address.

A replication partner can be a *pull* or a *push partner*. A pull partner is a WINS server that requests new WINS database entries (replicas) from its partner. This is done by requesting entries with a higher version ID than the last entry received from a partner during the last replication. The pull occurs at configured time intervals or in response to an update notification from a push partner. A push partner is a WINS server that sends update notification messages (If configured to do so by setting the update count or by turning on replicate up on address change. When it is not configured in this way, the WINS server will propagate triggers received from a partner to all other partners to its partner when its WINS database has changed. The partner then may pull these changed entries. The update notification occurs after a configurable number of changes to the WINS database.

When replication is configured between two WINS servers, it is recommended that both servers be push and pull partners of the other. The Primary and Secondary WINS server of any client **must** have a push and pull relationship with each other. In general, NetBIOS names are presented as a flat namespace. Attempts to impose a hierarchy on this namespace through one-way replication schemes result in problems with name uniqueness and resolution and cannot be recommended.

Overview of the Replication Process

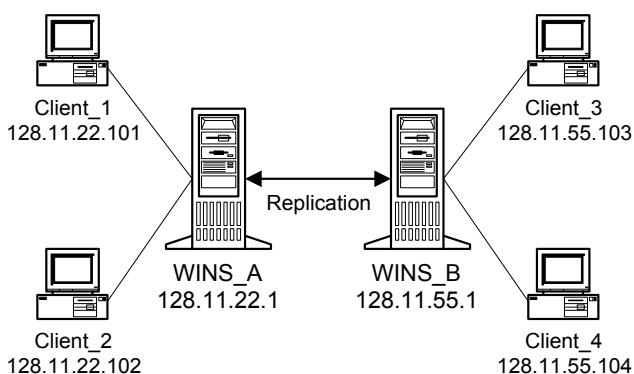


The figure above shows an extremely large WINS implementation—servicing in excess of 100,000 nodes. In a configuration with so many WINS servers it is tempting to create many push/pull relationships for redundancy. This can lead to a system that, while functional, is overly complex and difficult to understand and troubleshoot. We have imposed some order on the above configuration by using a hub structure. Four major hubs are located in Seattle, San Francisco, Chicago, and Los Angeles. These hubs serve as secondary WINS servers for their regions while connecting the four geographic locations. All primary WINS servers are configured as Push/Pull partners with the hubs and the hubs are configured as push/pull partners with other hubs. Let's assume the primary WINS servers replicate with the hubs every 15 minutes, and the hub to hub Replication Interval is 30 minutes. The convergence time of the WINS system is the time it takes for a node registration to be replicated to all WINS servers. In this case the longest time would be from a Seattle primary to a Chicago primary. The convergence time is 15 + 30 + 30 + 15 minutes or 1.5 hours. Of course we've ignored the fact that some of our WINS servers are connected across slow links. It is probably not necessary for the servers in Paris or Berlin to replicate every 15 minutes. We might configure them to replicate every two hours or even every 24 hours depending on the volatility of the WINS system. We will discuss more about replication across slow links in a later section. We mentioned redundancy earlier. In our configuration there is some, but not a lot of, redundancy. If the link between Seattle and Los Angeles is down,

replication will still occur through San Francisco. But what happens if the Seattle hub itself goes down? In this case, the Seattle area will no longer be able to replicate with the rest of the WINS system. Network connectivity, however, is still functional and all WINS servers contain the entire WINS database. Name resolution will continue to function normally. All that is lost are changes to the WINS system since the Seattle hub went down. A Seattle user won't be able to resolve the name of a file server in Chicago that came on-line while the Seattle hub was down. Once the hub is returned to service, all changes will be replicated.

The following example shows replication between two WINS servers in more detail.

Detailed Replication Example



Let's look at the database tables for WINS_A and WINS_B on January 1, 1996. All four clients were powered on this morning between 8:00 AM and 8:15 AM. Client_2 has just been shut down. The following parameters are set in WINS_A and WINS_B:

- WINS_A and WINS_B are push/pull partners to each other
- The Replication Interval is 30 minutes
- The Renewal Interval is 4 days
- The Extinction Interval is 4 days
- The Extinction Time-out is 1 day
- The Verify Interval is 24 days

Before replication, WINS_A has the following two entries:

Name	Address	Flags	Owner	Version ID	Time Stamp
Client_1	128.11.22.101	unique,active,h-node,dynamic	WINS_A	4B3	1/5/96 8:05:32 AM
Client_2	128.11.22.102	unique,released,h-node,dynamic	WINS_A	4C2	1/5/96 8:23:43 AM

WINS_B has the following two entries:

Name	Address	Flags	Owner	Version ID	Time Stamp
Client_3	128.11.55.103	unique,active,h-node,dynamic	WINS_B	78F	1/5/96 8:11:12 AM
Client_4	128.11.55.104	unique, active,h-node,dynamic	WINS_B	79C	1/5/96 8:12:21 AM

Clients_ 1, 3, and 4 were Time Stamped with Current Time + Renewal Interval at

the Time they booted and Client_2 was Time Stamped with Current Time + Extinction Interval when it was released. The version IDs indicate the value of the registration counter at the time of registration. The registration counter is incremented by one each time it is used to generate a new version ID in the database. Each WINS server has its own registration counter. The version ID jumps from 4B3 for Client_1 to 4C2 for Client_2. This indicates that fourteen registrations (or extinctions or release to active transitions) took place between the registration of Client_1 and Client_2.

Replication takes place at 8:30:45 by WINS_A's clock. WINS_B's clock is 8:31:15 at this time. Of course the replication won't all take place in the same second, but we'll use these times to generate the Time Stamps. Note that replication does not mean both pull at the same time. Each will pull according to its individual pull schedule. After replication, the WINS_A database will look like the following:

Name	Address	Flags	Owner	Version ID	Time Stamp
Client_1	128.11.22.101	unique,active,h-node,dynamic	WINS_A	4B3	1/5/96 8:05:32 AM
Client_2	128.11.22.102	unique,released,h-node,dynamic	WINS_A	4C2	1/5/96 8:23:43 AM
Client_3	128.11.55.103	unique,active,h-node,dynamic	WINS_B	78F	1/25/96 8:30:45 AM
Client_4	128.11.55.104	unique, active,h-node,dynamic	WINS_B	79C	1/25/96 8:30:45 AM

After replication, the WINS_B database will look like the following:

Name	Address	Flags	Owner	Version ID	Time Stamp
Client_1	128.11.22.101	unique,active,h-node,dynamic	WINS_A	4B3	1/25/96 8:31:15 AM
Client_3	128.11.55.103	unique,active,h-node,dynamic	WINS_B	78F	1/5/96 8:11:12 AM
Client_4	128.11.55.104	unique, active,h-node,dynamic	WINS_B	79C	1/5/96 8:12:21 AM

Client_1 has been replicated to WINS_B and Client_3 and Client_4 have been replicated to WINS_A. The replicas have all kept their original owner and version ID and have been Time Stamped with Current Time + Verify Interval.¹⁵ Client_2 has not been replicated because it is in the released state. This is a little unusual (but possible) because Client_2 shut down before its first replication. If Client_2 had not been shut down until after the replication, WINS_B would have a replica of Client_2 in the active state. This replica would remain in the active state even after Client_2 released because the change in state would not be replicated.

If Client_2 remains shut down for the Extinction Interval) it will be placed in the extinct (or tombstone) state. At the first scavenging after January 5, 1996 8:23:43 AM (assuming an Extinction Interval of four days), WINS_A's will look like the following:

¹⁵ The Owner ID value may be different between WINS servers because it comes from the IP address - Owner ID mapping table local to each WINS server. For example WINS_B will have the Owner ID 0 on WINS_B but the Owner ID 1 on WINS_A. Owner ID 0 is always used for marking itself.

Name	Address	Flags	Owner	Version ID	Time Stamp
Client_1	128.11.22.101	unique,active,h-node,dynamic	WINS_A	4B3	1/9/96 6:35:26 AM
Client_2	128.11.22.102	unique,tombstone,h-node,dynamic	WINS_A	657	1/6/96 9:50:53 AM
Client_3	128.11.55.103	unique,active,h-node,dynamic	WINS_B	78F	1/25/96 8:30:45 AM
Client_4	128.11.55.104	unique, active,h-node,dynamic	WINS_B	79C	1/25/96 8:30:45 AM

Notice that Client_2 has entered the tombstone state and that both the Time Stamp and the version ID have changed. The Time Stamp is now Current Time + Extinction Time-out and the new version ID will cause this entry to be replicated at the next replication. Notice also that Client_1 has a new Time Stamp while retaining its version ID. It has continued to be renewed throughout the last four days. The renewal rate is dependent on the client stack.

After replication at 10:00:23 AM WINS_B's database will look like the following (notice that Client_3 and 4 were renewed):

Name	Address	Flags	Owner	Version ID	Time Stamp
Client_1	128.11.22.101	unique,active,h-node,dynamic	WINS_A	4B3	1/25/96 8:31:15 AM
Client_2	128.11.22.102	unique,tombstone,h-node,dynamic	WINS_A	657	1/6/96 10:00:23 AM
Client_3	128.11.55.103	unique,active,h-node,dynamic	WINS_B	78F	1/9/96 8:11:12 AM
Client_4	128.11.55.104	unique, active,h-node,dynamic	WINS_B	79C	1/9/96 8:12:21 AM

If Client_2 remains down for one more day (the Extinction Time-out) it will be removed from the databases at scavenge time.

The WINS_A database will look like the following:

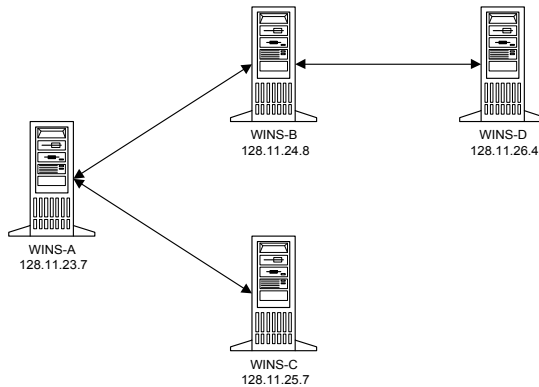
Name	Address	Flags	Owner	Version ID	Time Stamp
Client_1	128.11.22.101	unique,active,h-node,dynamic	WINS_A	4B3	1/11/96 9:45:56 AM
Client_3	128.11.55.103	unique,active,h-node,dynamic	WINS_B	78F	1/25/96 8:30:45 AM
Client_4	128.11.55.104	unique, active,h-node,dynamic	WINS_B	79C	1/25/96 8:30:45 AM

The WINS_B database will look like the following:

Name	Address	Flags	Owner	Version ID	Time Stamp
Client_1	128.11.22.101	unique,active,h-node,dynamic	WINS_A	4B3	1/25/96 8:31:15 AM
Client_3	128.11.55.103	unique,active,h-node,dynamic	WINS_B	78F	1/11/96 9:44:27 AM
Client_4	128.11.55.104	unique, active,h-node,dynamic	WINS_B	79C	1/11/96 9:46:44 AM

During the first scavenging after January 25, 1996 8:30 AM WINS_A will verify with WINS_B that Client_3 and Client_4 are still valid active names. WINS_B will do the same for Client_1 with WINS_A.

Pulling WINS Database Entries



The WINS server maintains an IP address - Owner ID mapping table in its local database. This table gives the mappings between the IP addresses and Owner IDs of WINS servers that have entries in its local database. A sample IP address - Owner ID mapping table for WINS-A is given below

IP Address	Owner Id
128.11.23.7	0
128.11.24.8	1
128.11.25.7	2

During WINS initialization the WINS server scans the Name to IP address mapping table to determine the maximum version ID corresponding to each owner registered in its database. It creates an in-memory table (this table is never committed to the database), called the Push Partner -Version # mapping table and initializes it with the information retrieved above. This table has an entry for each Push partner. Each entry contains the maximum version ID found for all owners in the local database of the Push partner. For example, if the local WINS server has an Owner ID of 0, then given the above IP-Address to Owner ID mapping table, the Push Partner-Version # table might look like the one given below:

	0	1	2
0	100	900	630
1			
2			

It shows that at the local database of the WINS server identified by Owner ID 0, there are entries owned by three WINS servers with Owner IDs of 0, 1, 2. The highest version numbers for entries are 100, 900, and 630, respectively.

The WINS server is now ready to determine whether it needs to update its database. It sends a message to each push partner it has asking it to respond back with the IP address-max version number records pertaining to its local database. As it hears back from each push partner, the WINS server fills and grows its own table.

The table might take on more columns. For example, if WINS_B at IP address 128.11.24.8 (Owner ID 1) responds back with a record for WINS_D at IP address 128.11.26.4, a column for an Owner ID 3 is added to the local Push Partner - Version Number mapping table. This column is for the indirect push partner, WINS_D, at IP address 128.11.26.4. At the same time the mapping 128.11.26.4 - 3 is stored in the IP address-Owner ID table. The relevant cells in the new table are initialized. To illustrate by example, let us say that WINS_B at 128.11.24.8 responded back with the following three records:

```
128.11.24.8 999
128.11.26.4 700
128.11.23.7 89
```

The local Push Partner - Version Number table would now become:

	0	1	2	3
0	100	900	630	0
1	89	999	0	700
2				

The IP address to Owner ID table would be grown by the following record

128.11.26.4	3
-------------	---

After all Push partners have responded, the *IP address-Version #* table should be fully filled up.

	0	1	2	3
0	100	900	630	0
1	89	999	0	700
2	93	879	820	0

It is examined to determine which push partner has the latest data for each owner. A WINS server will always have the highest version ID for entries owned by it, however, all WINS servers might not be partners of the requesting pull partner. The starting version ID that is required to sync the local database is then determined and the push partner is requested to send the database records with version IDs equal or greater than this. The push partner will send all records with version numbers higher than that. If a push partner has the latest data for more than one owner, one single request may be sent to retrieve the records for all. Of course, in this simple example, who has the most current data for a database never changes. When WINS_A pulls data, WINS_B has the latest data for itself and WINS_D; WINS_C has the latest data for itself. In a more complex model such as the one shown earlier in the figure at the beginning of this section, there are loops in the replication paths and replication takes place at differing intervals.

When the push partner receives a request from another WINS server it retrieves the required records from its local database and sends them in the response. The

records are retrieved by seeking to the record that starts the range and then moving sequentially over the records until the last one in the range has been retrieved.

When the data is received from the push partners, the pull partner updates its database.

What gets replicated As described above, all entries with version IDs greater than those in the pulling database get replicated. However, not every change to a database causes the version id of a record to be incremented.

Records in the WINS database contain state and ownership information. Records may be in an active, released, or extinct (tombstone) state. They are owned by the local database or are replicas from another WINS server. A record is also static or dynamic.

When names are registered with a WINS server, they are entered in the database in an active state and Time Stamped with Current Time + Renewal Interval. The version ID is taken from the version id counter which is then incremented. If a name is explicitly released or fails to refresh during the Renewal Interval, it enters the released state. The entry is Time Stamped with Current Time + Extinction Interval. The version ID is not changed. Thus, records in the released state will not be replicated.¹⁶ If a record remains in the released state for more than the Extinction Interval, it enters the extinct (or tombstone) state. It is Time Stamped with Current Time + Extinction Time-out and receives a new version ID to cause it to be replicated. If a record remains in the extinct (or tombstone) state for more than the Extinction Time-out, it is deleted from the database.

Only records in the active or tombstone states are replicated. In the replica database these records are entered with the fields received from the owner database with the exception of ownership and Time Stamp. The ownership field is taken from the local IP address - Owner id mapping table. This does not mean that the ownership of the replica changes. It just means that the value used to represent a particular WINS server differs from server to server. For example, WINS_D might be represented by a 2 on WINS_B but a 3 on WINS_A. An active record is Time Stamped with (local) Current Time + Verify Interval. A tombstone record is Time Stamped with (local) Current Time + Extinction Time-out.

Conflicts Detected During Replication

Name conflicts are normally handled at the time of name registration. However, it is possible for the same name to be registered at two different WINS servers. This would happen if the same name were registered at a second WINS server before the database from the first WINS server had been replicated. In this case, the conflict will be noted and resolved at replication time.

Conflict at replication can be:

- Between two unique entries
- Between a unique entry and a group entry
- Between two group entries

A multihomed entry is a unique entry with multiple addresses.

¹⁶ Records in the released state will **never** be replicated, even if their version IDs falls within the to be replicated range.

Conflict between unique entries When resolving conflicts between unique entries the following is taken into account:

- **State of the entries.** The database entry can be in the active, released, or tombstone state. The replica can be either in the active or tombstone state.
- **Ownership of the entries.** The WINS server may or may not own the database entry.
- **Addresses of the entries.** The addresses of the entries may or may not be the same.

Conflict between two replicas with the same or different IP addresses The replica in the database is overwritten by the new replica, regardless of whether the addresses match or not, unless the replica in the database is active and the new replica is a tombstone. In this case, the replica in the database is not overwritten by the new replica unless they are both owned by the same WINS server.

Conflict between owned entry and replica with same IP address The database record is replaced by the replica unless the database record is active and the replica is a tombstone. In that case, the version ID of the database record is incremented so that the record gets propagated at replication time.

Conflict between owned entry and replica with the different IP addresses

The database record is replaced by the replica unless the database record is active. In such a case, if the replica is a tombstone, the database record's version ID is incremented so that the record gets propagated at replication time. If the replica is also active, then the database record's node is challenged to determine if it still has the name. If it still has it, the replica's node is sent a Name Conflict Demand forcing the node to place the name in the conflict state.

Conflict between a unique entry and a group entry When the conflict is between a unique entry and a group entry, the group entry is kept. If the unique entry is owned by the WINS server and is not in the released or tombstone state, the WINS server asks the node in the unique entry to release the name.

Conflict between two Special Group entries The database record is replaced by the replica unless it is active. If it is active, its version ID is incremented so that the record gets propagated at replication time. If the replica is also in the active state, the member list of the database record is updated with any members in the replica that are not already in it. If the list of members in the active state grows to more than 25, the extra members are not inserted.

Conflict involving a Multihomed record If a multihomed replica conflicts with a tombstone or released entry in the database, the entry in the database is replaced with the replica unless the entry is a normal group and is in the released state. This is no different from the other scenarios where a non-multihomed entry conflicts with a released normal group entry.

If a tombstone multihomed replica clashes with an active database entry whose owner is the same as that of the multihomed replica, then the database entry is replaced. If however, the active database entry is a replica owned by a different owner, it is not replaced. If the active database entry is owned by the local WINS server and happens to be a unique entry, its version ID is incremented to cause propagation.

If an *active* multihomed replica clashes with an *active* unique/multihomed replica in the local database having the same owner, it is replaced. If the owner is different, it is not replaced. If the entry in the database is owned by the local WINS server, and if the members of the record (one the in case where it is a unique record) is a subset of the members in the replica, the database record's Time Stamp is changed and version ID incremented to propagate it. If the replica's members are not a subset, the local record's addresses are challenged. If all challenges succeed (that is the node challenged does not respond to any challenge), the database record is replaced. If at least one challenge fails, the node is told to release the name from all addresses prior to replacing the database record with the replica.

If a multihomed replica clashed with an active group entry in the database, the version ID of the entry in the database is incremented to cause propagation.

If a non-multihomed replica clashes with a non-active multihomed record in the database, the database record is replaced. If it clashes with an active multihomed entry in the database owned by the same owner, the database record is replaced. If the multihomed entry in the database is a replica owned by a different owner, it is not replaced. If however, it is owned by the local WINS server and the replica pulled in is a unique record, then the addresses in the multihomed record are challenged. If all challenges succeed, the database record is replaced. If at least one challenge fails, the addresses in the database record are sent release requests for the name and then the database record is updated.

Note: The address in the unique replica, if present in the member list, is ignored in the above situation.

Scavenging

Scavenging maintains the correct state information in the database. The Scavenging Timer is started at boot time and is equal to $\frac{1}{2}$ the Renewal Interval. Scavenging first occurs after $\frac{1}{2}$ the Renewal Interval.¹⁷ During the first scavenging, all scavenging actions are performed except for the deletion of the tombstones. Tombstones may not be deleted until at least three days have elapsed since boot. This is to allow for sufficient time for their replication. Scavenging reoccurs at $\frac{1}{2}$ the Renewal Interval (or may be initiated manually).

Scavenging follows the following algorithm:

¹⁷ If the Renewal Interval is changed before the first scavenge, then scavenging would first occur after $\frac{1}{2}$ of the new Renewal Interval from the time it was changed.

```
Get records owned by self
If Current Time > Time Stamp
    Change State
        Active -> Released
        Released -> Tombstone
        Tombstone -> delete from database
Get replica Tombstones
If Current Time > Time Stamp
    Delete record from database
Get Active replicas
If Current Time > Time Stamp
    Verify with owner that record still exists
    If exists
         $Time\ Stamp = Current\ Time + Verify\ Interval$ 
    Else
        Delete record from database
```

NEW IN NT 4.0

Burst Mode

The burst handling parameter is used to temporarily achieve a steady state in the WINS server when the WINS server is either started with a clean database or many WINS clients come online for the first time. Either situation causes a large amount of name registration and name refresh traffic to occur. The WINS server currently stores a maximum of 25000 name registrations and refresh queries in its queue before dropping queries. By using this parameter the WINS server can be configured to send success responses to the clients whose requests are dropped. These responses will have TTLs that will slow down the refresh rate of those WINS clients and regulate the burst of WINS client traffic. This will result in a steady state being reached much more quickly.

This feature may be enabled by creating the "BurstHandling" key under the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WINS\Parameters key and setting the value to 1.

Name: BurstHandling

Type = REG_DWORD
Value: 0 or 1
Default = 0

Administering WINS Through a Firewall

When doing remote WINS administration, an initial session set up is to port 135. This is followed by another session to a random port above 1024. This is because the WINS Administrator uses "dynamic endpoints" in RPC. Internet firewalls cannot be configured to pass this traffic when the port is not consistent. In Windows NT 4.0 the system defaults for dynamic port allocation may be defined in the registry.

In order to allow remote administration of WINS through a firewall, a list of all ports available (or not available) from the Internet should be defined in the registry with the following keys. These keys are located under:

HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc\Internet

Name: Ports

Type: REG_MULTI_SZ - set of IP port ranges

Description: Specifies a set of IP port ranges consisting of either all of the ports available from the Internet or all of the ports not available from the Internet. Each string represents a single port or an inclusive set of ports (for example, "1000-1050" "1984"). If any entries are outside the range of zero to 65535, or if any string cannot be interpreted, the RPC run time will treat the entire configuration as invalid.

Name: PortsInternetAvailable

Type: REG_SZ - Y or N (not case-sensitive)

Description: If Y, the ports listed in the Ports key are all the Internet-available ports on that machine. If N, the ports listed in the Ports key are all those ports that are not Internet-available.

Name: UseInternetPorts

Type: REG_SZ - Y or N (not case-sensitive)

Description: Specifies the system default policy. If Y, the processes using the default will be assigned ports from the set of Internet-available ports, as defined

above. If N, processes using the default will be assigned ports from the set of intranet-only ports.

Consistency Check by WINS

In Windows NT 4.0 it is possible to periodically check the WINS database for consistency. However, consistency checking is very network intensive and consumes a lot of cycles on the WINS server. This is because the WINS replicates all records for an owner whose records are being checked from another WINS to determine whether its database with regard to the owner is in synch with the other WINS. One should be prudent in selecting the values for different parameters below. The type of existing network configuration (number of WINS servers, WAN/LAN lines between WINS servers, number of WINS clients, and so on) will help to determine appropriate values for these parameters.

This feature may be enabled by creating the "ConsistencyCheck" key under the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WINS\Parameters key.

Optionally, the following values may be created under this key

Name: TimeInterval

Type: REG_DWORD - No Of Seconds

Description: Specifies the time interval at which WINS will do a consistency check.

Default: 24 hours

Name: SpTime

Type: REG_SZ hh:mm:ss

Description: Specifies the specific time in hh:mm:ss format at which the first consistency check will be done. Thereafter, it will be done periodically at TimeInterval seconds.

Default: 2:00:00 (2 AM).

Name: MaxRecsAtATime

Type: REG_DWORD No Of Recs

Description: The maximum number of records that will be replicated in one consistency check cycle. WINS does consistency checks on the records of each owner of WINS. After checking one owner it either goes on to the next one in its list of owners or stops based on the MaxRecsAtATime value.

Default: 30000

Name: UseRplPnrs

Type: DWORD 0 or non-zero value

Description: If set to a non-zero value, WINS will contact only its pull partners when doing consistency checks on the records in its database. If the owner whose records need to be checked is a Pull Partner, it will be used. Otherwise, a random pull partner will be used.

Note: WINS never deletes records in its database if the partner with which it is doing verification is not the owner. This is because WINS does not know which database is more current.

WINSCHK

There is a new tool in the Windows NT 4.0 Resource Kit—WINSCHK. This

command-line utility checks name and version number inconsistencies that may appear in Windows Internet Name Service (WINS) databases, monitors replication activity, and verifies the replication topology in an enterprise network. It is particularly useful for WINS administrators.

With WINSCHK, you can remotely check and resolve WINS database replication issues. You can isolate some of the most common problems that lead to database inconsistencies with an easy-to-use tool that can be run in a central location.

This tool supplements WINSCL with options geared towards flagging possible causes for database inconsistencies, such as:

- asymmetric replication topologies
- high communication failures (WINSCHK can detect these and warn the administrator)

The tool also helps monitor the actual replication activity by allowing the user to:

- check for version number inconsistencies
- check for the state of one or more names at various WINS databases in the network.

The tool can be used in interactive or non-interactive mode. In the latter, a log of all the tool activity is kept in the local directory (WINSTST.LOG). The option to monitor WINS activity can be run in the background, which dumps all logs into MONITOR.LOG.

WINSCHK provides the following options:

- 0 - Toggle the interactive switch (default value: Interactive)
- 1 - Test for names (in names.txt) against WINS servers (in servers.txt)
- 2 - Check version number consistencies
- 3 - Monitor WINS servers and detect communication failures
- 4 - Verify replication configuration setup

99 - Exit this tool

0 - Allows the user the option of having status messages printed on the command window. All status messages are logged into winstst.log and on the command window if Interactive switch is ON.

1 - Test for N names against M servers

This is a quick tool to check for consistency between various WINS servers. The utility is driven by two flat files which can be edited using your favorite text editor. The file "servers.txt" contains the IP address of a starting WINS server from which a list of all the replicating WINS servers is built up to query. The file names.txt contains a list of NetBIOS names to query and may contain multiple NETBIOS names (one per line) which need to be checked. The format of the name in this file is <name>*<16th byte>, for example. FOOBAR*20. The names in the file must be in uppercase. The utility will run the list of NetBIOS names querying each WINS server. It will check for consistency of addresses and report any occurrences of "name not found" or mismatched IP addresses. It will also report non-responsive WINS servers.

2 - Check version number inconsistencies

Get the owner address—version number maps (through an RPC function) from different WINS servers and check the consistency of their databases by ensuring that a WINS server always has the highest version number among the network of WINS servers for records owned by it.

Example:

```
A  B  C    <--- list of owners
A 100 80 79  <-- mapping table retrieved from A
B 95 75* 65  <---mapping table retrieved from B
C 78 45 110  <---mapping table retrieved from C
```

Intersection B with B indicates a problem and needs fixing.

3 - Monitor WINS servers and the communication failures between WINS servers.

This can be run in a one time or a continuous version. The continuous version kicks in every three hours by default. It is recommended that this option not be run too often to avoid excessive network activity. This option logs activity in Monitor.log.

Monitors WINS servers periodically to ensure that both the primary and backup are not down together. Also retrieves WINS statistics periodically, every hour for instance, to ensure that replication is not failing consistently. In either case, the administrator is alerted to the situation. The communication problems may be due to a disconnected WINS or one that is down.

4 - Verify replication configuration setup

Check the registry of a WINS server to ensure that each partner is pull and push and that a pull interval is defined. Check the same for each partner. In this way, the entire network is covered. If an asymmetric partner relationship is discovered it is flagged for the administrator.

WINS DATABASE

The WINS server uses a relational database engine to access an ISAM (Indexed Sequential Access Method) database. The WINS database consists of two tables. The IP address - Owner ID mapping table and the Name to IP address mapping table.

IP Address - Owner ID Mapping Table

This table contains a row for each WINS server that has entries in the Name to IP address mapping table. A row gives the mapping between the IP address of an WINS server and its identifier as stored in the Owner ID field of the entries owned by it.

An entry contains the following fields:

IP address

Type (4 bytes)	Length (4 bytes)	Value (number of bytes indicated by length)
----------------	------------------	---

The type field indicates the address family (TCP/IP, OSI, SPX/IPX, and so forth). Only TCP/IP is implemented. The length field indicates the number of bytes in the value field. The value field is the address of the node.

Owner Identifier

Owner Identifier
4 bytes (Windows NT 4.0) or 1 byte (Windows NT 3.51)

The Owner Identifier is found in the Owner ID field of all Name-IP address mapping table's records that were created/updated by the WINS server at this IP address.

Name to IP Address Mapping Table

This table stores the name to IP address mappings. The entries in this table are created as a result of name registration requests received from NetBIOS over TCP/IP nodes and replicas received from other WINS servers. The Name to IP address mapping table has two indices. There is a clustered index on the name field. This allows fast retrieval of records required for name queries. There is a *primary index* built from concatenation of the Owner ID and version ID fields, in ascending order. This allows fast access of records falling within ranges of version IDs for a particular owner.

An entry contains the following fields:

Name This is a text field that can contain names from 1-255 characters in width. NetBIOS names are 1-16 characters wide. A NetBIOS name with appended scope can be up to 255 bytes long.

Addresses This is a binary field (unlimited size) which stores the binary addresses corresponding to the name. Each address is of the form TLV (Type, Length, Value). The type field (4 bytes) indicates the address family (TCP/IP, OSI, SPX/IPX, and so on). The length field (4 bytes) indicates the number of bytes in the value field. The value field is the address of the node.

Unique Name Entry

This field contains just one address:

Type (4 bytes)	Length (4 bytes)	Value (number of bytes indicated by length)
----------------	------------------	---

Special Group Entry

The first 4 bytes give the number of address records that follow. Unlike a unique entry, a special group entry's address record is comprised of two additional components besides the TLV for the address. These two components, occurring before the addresses are Owner ID (4 bytes) indicating the owner of the address entry, and the Time Stamp (the time when the address entry was last refreshed). Each address record is of the following form:

Owner id (4 bytes)	Time Stamp (4 bytes)	Type (4 bytes)	Length (4 bytes)	Value (Number of bytes indicate by length)
-----------------------	-------------------------	-------------------	---------------------	---

Flags field

This is an unsigned byte field.

Bit 0 & 1 Record Type
 0 = unique
 1 = normal group
 2 = special group
 3 = multihomed

Bit 2 & 3 Record State
 0 = active
 1 = released
 2 = tombstone
 4 = Reserved

Bit 4 & 5 Client node type if entry is unique or multihomed
 0 = B-node
 1 = P-node
 2 = M-node
 3 = H-node

Bit 7 Dynamic/Static
 0 = dynamic
 1 = static

Owner id This is an unsigned 4 byte value (1 byte in Windows NT 3.51) that stores a unique ID identifying the owner of the entry. The owner of an entry is the WINS server that inserted/updated the entry. This field is used during replication to determine who registered/updated the entry. The Owner ID maps to an IP address. This mapping is given in the IP address to Owner Id mapping table.

Version ID This is an 8 byte field. Every entry inserted/updated (except state changes from active to the released state) by an WINS server is stamped with the

value of an WINS server registration counter. This registration counter is incremented by one for each registration/update. Because this is a 64 bit number, the possibility of rollover may be ignored. Even at the unreasonably high rate of 1000 registrations/second, rollover would not occur for $6 * 10^8$ years.

Time Stamp This is a date and time field which stores the time in seconds since midnight Jan. 1, 1970 (UCT). This field is long (4 bytes). This field is used to determine how old the entry is at database cleanup times.

Winsdmp

Winsdmp is a command line interface tool to dump the WINS database. It dumps a comma separated list of entries to stdout. This tool is part of the Windows NT Resource Kit. The following CSV fields are output by Winsdmp:

- Owner IP Address,
- Name,
- 16th character of name in hexadecimal
- Name length,
- Type of record (unique/multihomed/special group/normal group),
- State of record (active/released/tombstone),
- Version ID (high order word),
- Version ID (low order word),
- Static/Dynamic flag,
- Time stamp,
- Number of IP addresses,
- IP addresses

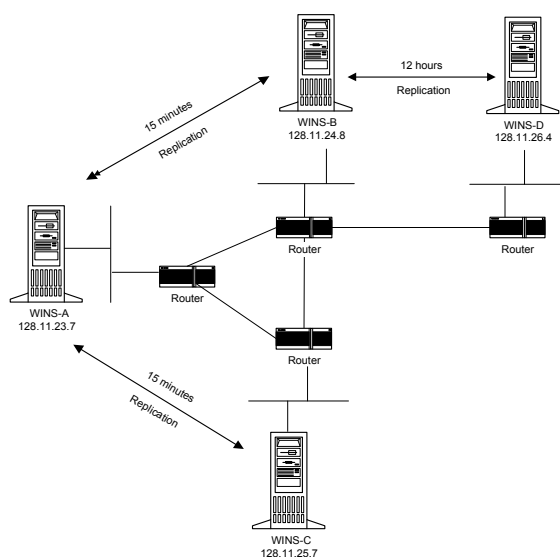
DESIGNING A WINS INFRASTRUCTURE

Practical Consideration for Design

In an enterprise network environment NetBIOS connectivity is needed on a routed network which often spans the globe. The NetBIOS name space is flat, each name must be unique and there must be a mechanism to convert the NetBIOS name to an address. WINS is the Microsoft implementation of a NetBIOS name server as described in RFC 1001 and RFC 1002. WINS implements a distributed database for NetBIOS names and their corresponding addresses. WINS clients register their names at a more or less local WINS server and the WINS servers will replicate the entries (NetBIOS name/IP address pairs) to the other WINS servers. This mechanism ensures uniqueness of the NetBIOS names and makes local name resolution possible.

Design criteria

Convergence time A consideration of each design is the convergence time of the WINS system. This is defined as the worst-case time it takes, for a specific configuration, to get a new entry in a WINS server database replicated to all other WINS databases. A configuration guarantees that name queries for a new name will succeed after this convergence time. Before that clients may or may not be able to find this new (or just modified) machine. This is demonstrated in the following figure. A client registers its name in WINS_C. Other clients can query WINS_C for this name and will get the IP address. Clients who query the other WINS servers (A, B and D) at this moment will not get a positive response to their query until the entry is replicated to A, B and D. Replication from WINS_C to WINS_A will take place when the push update count threshold (as configured on C) is exceeded or when the pull Replication Interval (15 minutes, as configured on A) expires. We can only guarantee that the entry will be replicated when the pull Replication Interval expires. Queries for the new name to WINS servers B and D may still not be successful. Another 15 minutes later we can guarantee that the entry will have been replicated to WINS server B, and 12 hours later we can guarantee that it will have been replicated to WINS server D. The convergence time of this configuration is twice 15 minutes plus 12 hours is 12.5 hours. Name query requests may, in reality, succeed before the convergence time has passed. This would happen if the entries would have to be replicated over a shorter path than the worst case path. And it would also happen when an update count Threshold would be passed before the Replication Interval would expire, that would result in earlier replication of the new entry. The longer the replication path, the longer the convergence time. The pull replication Intervals in this example are just an example. Fifteen minutes between sites is relatively short, 12 hours is very long (even between continents). The best choice for these intervals depends on the requirements. The convergence time of a configuration is a result of the design choices.



Fault tolerance

There are two basic types of failures:

- A WINS server may crash or be stopped to do maintenance
- Network failures: links go down or routers may fail

In the diagram above, a failure of WINS-A or WINS-B would segment the WINS configuration. Entries would no longer be replicated from WINS-C to WINS-D and vice versa. Because the IP address and name no longer match for updated clients, other clients would not be able to connect to the updated machines. Adding replication between WINS-B and WINS-C would improve the configuration for cases in which WINS-A fails. Adding replication between WINS-D and WINS-C would improve the configuration for WINS-B failures.

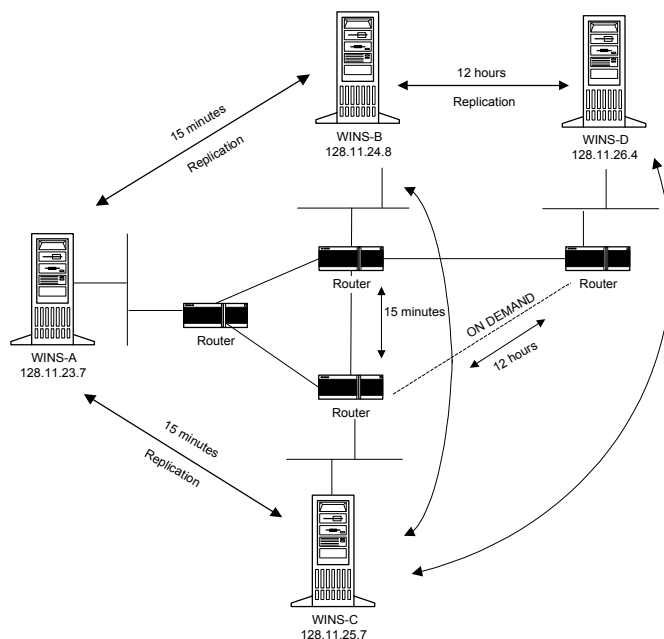
Failures of one of the links between A, B, and C wouldn't hurt the WINS configuration because the underlying router network would reroute the traffic. Although this may not be the most efficient situation when compared with a fully operational network, WINS replication will continue relatively undisturbed. Failure of the link between B and D, however, would segment the WINS configuration. Because this would also make other network traffic impossible there should be an on demand backup link between D and C. The WINS replication traffic would then be rerouted by the underlying router infrastructure.

In the diagram above, the routers are all single points of failure. When one of them fails, it would segment the WINS configuration. A generally accepted approach is to look at two simultaneous failures and their consequences.

A segmented WINS configuration is not as dramatic as it may sound. Clients can usually still resolve the names to addresses. Local WINS servers and/or broadcasts will take care of most of the name resolution. Only updated or new remote entries are unknown. Entries are not dropped at scavenging time when the owning WINS cannot be reached. A WINS service can be installed on another machine when the hardware of the regular WINS server machine fails. The database backups can be restored on the new machine.

Duplicate replication traffic Fine tuning the Replication Intervals may

save some bandwidth on WAN links. When we add the improvements to the previous diagram as mentioned earlier we get the diagram below. The new configuration is not only more fault tolerant, but it also has a shorter convergence time. In the previous figure, the longest path was from C over A and B to D. Now the longest path is from A or C over B to D. The new convergence time is now 12 hours and 15 minutes.



By keeping the pull Replication Intervals between WINS-C and WINS-B short (15 minutes), WINS servers A, B and C are always reasonably well synchronized. Replicas are never pulled in twice, only replicas with higher version IDs are copied. When WINS-B has an entry directly from WINS-C it will not pull that replica again from WINS-A. But we do want to avoid both WINS-D and WINS-B pull replicas from WINS-C over the link between B and C. There is a chance that this may happen. Suppose WINS-B pulls the replicas from WINS-C and then WINS-D pulls replicas from WINS-C. Then the load on the link between B and C would be higher. That would be avoided if WINS-D pulls them from WINS-B first and then checks WINS-C. The pull replication Interval between WINS-D and WINS-C would typically be the same 12 hours. Remember to configure push update counts (on WINS servers D and C) which correspond with the 12 hours pull Replication Interval, otherwise unexpected replication will be triggered by the update count threshold and not by the pull Replication Interval.

Server size

Database size

The effective size of the each connected WINS server database is roughly identical. Each database will have the same number of entries (neglecting latencies). The real size may be much larger because unused space is not reclaimed efficiently. Therefore the database has to be compacted regularly. In

Windows NT 3.5x this is accomplished with the jetpack.exe utility. The WINS implementation of

Windows NT 4.0 has dynamic compacting.

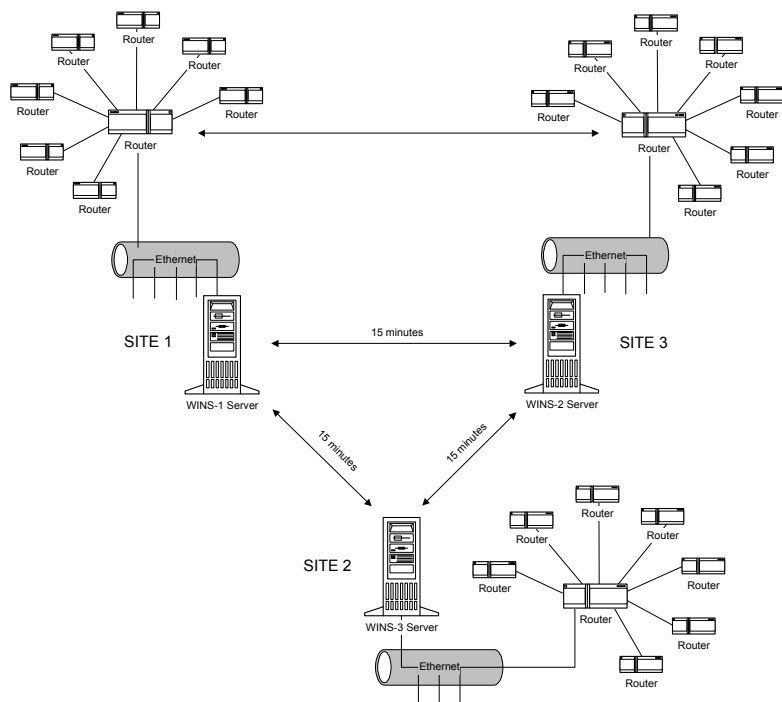
The size of the database is proportional to the number of entries. Unique entries will typically occupy 42 bytes (no scope ID) and Internet Group entries may occupy as many as 25 addresses and therefore more bytes.

Server Performance The sustained rate of a WINS server is the rate at which the WINS server can handle a continuous flow of requests. For a stand-alone WINS server we can rely on 1500 registrations per minute and 4500 queries per minute. This is not a number which will be reached easily in normal configurations.

Another rule of thumb is to have one WINS server (+backup) per 10,000 users. (This corresponds approximately with 10,000 clients who each register 4 names in half an hour, that is 1300 registrations per minute. This is not an unlikely scenario. In many businesses a large number of employees will arrive at work and turn on their machines within a half an hour.)

Some configuration examples

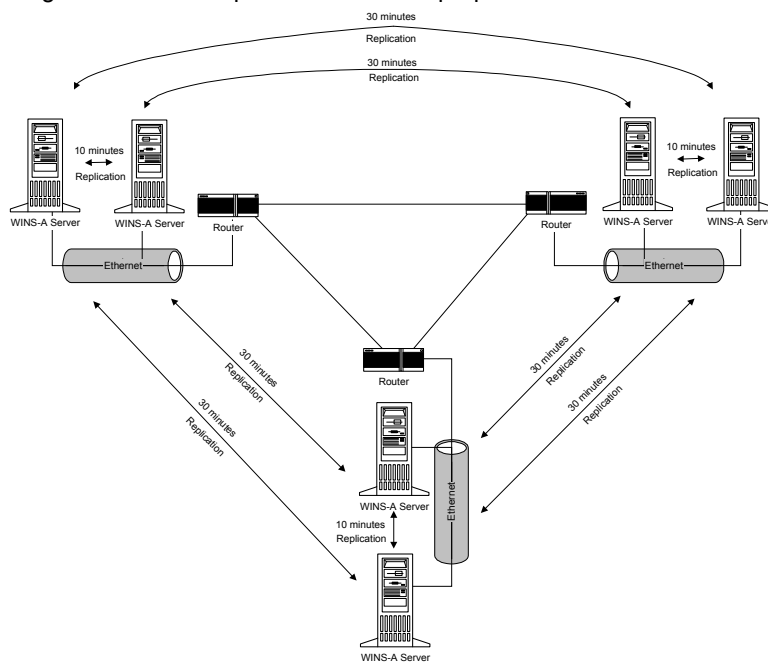
A medium-sized company, see figure below, has two main sites (1 and 3) with 500 PCs each and are connected through relatively high-speed links. It also has over 160 small branches. To save on the costs of the links there are some branches which act as concentrators for a region (like site 2).



The branches may have local servers, but usually they don't. When a link fails, local names can still be resolved by the broadcast mechanism. There is no need to put a separate WINS server in all of the branches. When the costs of registration and query traffic increase above the cost of a regional WINS server then regional

WINS servers will be used. The regional WINS servers are not required for correct functioning of the configuration, it is only a cost optimization. Actually they should be avoided whenever possible because they increase the convergence time. Regional WINS servers (like Site 2) are configured as replication partners of the WINS servers in the main sites (1 and 3). Clients in the main site (and those in the directly connected branches) are configured with the IP address of their local WINS server as primary and the IP address of the WINS server in the other main site as secondary. Clients in the branches connected to a regional concentrator will have the IP address of the regional WINS server as primary and the address of the WINS server in the closest main site as their secondary.

Another company, shown in the figure below, is very different. It is a bit larger company with 3 sites, each with 5000 users. The sites are connected with multiple T1 links. The number of users per site justifies a primary and secondary WINS server per site. Clients in the sites will be configured with a local primary and secondary WINS server. Half of them will have one local WINS server as primary and the other as secondary. The other half of the clients will have exactly the opposite configuration where the primary and secondary WINS server are concerned. This balances the registration and query load over both WINS servers, and gives a hot backup for maintenance purposes and in case of a calamity.



The local WINS servers are working with a very short pull Replication Interval, 10 minutes. This means that machines within the same building will be reachable within these 10 minutes. The Replication Interval between the sites can be larger; say 30 minutes. In general most users will work with resources on their local servers. In case of emergency an Administrator can always use the Send Replication Trigger Now (push/pull) to synchronize after some incident.

Network Traffic

The performance of a WINS system will depend on other traffic in the network. When the WINS server is not on the local (sub)network, but somewhere in the WAN, then requests and responses will have to go through the queues of the routers, which will cause delays at peak times. Bulk transport during replication will always get its fair share of the network bandwidth, which may be considerably lower than the theoretical bandwidth.

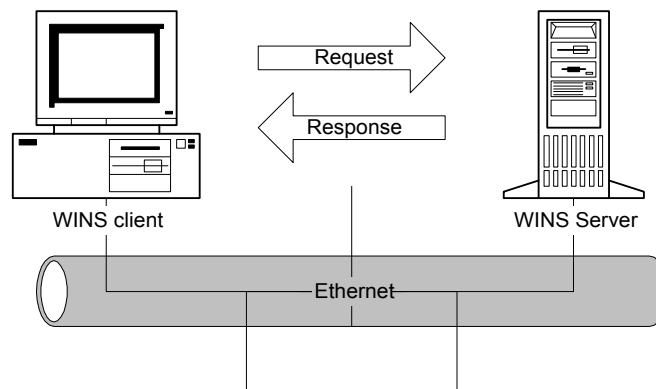
The messages as described below are UDP/TCP messages. Before they can be sent, the physical address has to be known. When the destination is on the same subnetwork, that means that an ARP will have to be sent. When it is on another subnet, then an ARP may be required to get to the router. This depends on the contents of the ARP cache. When the IP address is already in the ARP cache, there will be no new ARP message. Registrations are usually done in groups. Only one ARP per group will be required and this is not WAN traffic. Therefore ARP will not be mentioned in the discussion below.

All message sizes will be for the messages without a scope ID. The message sizes described here are for Ethernet, on another medium (Token-Ring, FDDI, WAN, and so on) the headers (and therefore the total length) are a bit different.

Typical network traffic

WINS server and WINS client traffic

A Name Registration request is sent for every NetBIOS name used by an application. The registration happens when the application (often implemented as a service) for a name starts, which is usually at boot time. For a client, the minimum number of names are the computer names (actually two, one for the workstation component <00> and one for the messenger <03>); the Domain name; and one for the user (messenger name <03>). For a server there will usually be more names which may include servername (computername but last byte <20>), more variants of the domain name (for browsing <1B>, <1D> and for the Domain Controllers <1C>), replicator account, Microsoft Systems Management Server account and so forth. The Name Registration request packet is 110 bytes. A positive Name Registration response is 104 bytes.



Request: NBT NS: MultiHomed Name Registration req. for MCSPAULLEM2 <00>

Response: NBT NS: Registration (Node Status) resp. for MCSPAULLEM2 <00>, Success, Owner Addr. 10.0.0.18

A Name Release request is sent for a name when its service stops, typically when the system shuts down. The Name Release request is 110 bytes and the Name Release response is 104 bytes.

Request: NBT NS: NS: Release req. for MCSPAULLEM2 <00>

Response: NBT NS: Release (Node Status) resp. for MCSPAULLEM2 <00>, Success

A Name Refresh request (renewal) is sent regularly while the name is registered. The request is 110 bytes, the response is 104 bytes. The time between renewals depends on the client implementation and the renewal interval. The NTW 3.51 implementation sends a Name Refresh requests after half the renewal interval to the primary WINS. When the primary goes down the renewals will be sent at the rate as specified by renewal interval of the secondary. Only half the renewal attempts are actually done at the secondary, the other attempts are sent to the primary. One (failing) renewal attempt at the primary is three packets when the WINS service at the primary is stopped.

Request: NBT NS: Refresh req. for MCSPAULLEM2 <00>

Response: NBT NS: Registration (Node Status) resp. for MCSPAULLEM2 <00>, Success, Owner Addr. 10.0.0.18

Name Query traffic depends on the application and the server. The application may disconnect from the server regularly to release the NetBIOS session. The file server may disconnect idle sessions. Different applications may connect to different servers. This all results in Name Query traffic.

Name Query request is 92 bytes, the response is 104 bytes.

Request: NBT NS: Query req. for DEL486NT

Response: NBT NS: Query (Node Status) resp. for DEL486NT, Success

Replication and verify traffic Replication and verify traffic are a bit more complicated, because they are performed in batches to reduce traffic. And they may sometimes result in challenge traffic. There is the basic load of a TCP connect and release. The load per unique entry is between 12 and 50 bytes. But, for instance, the Internet group entries depend on the number of registered nodes (25 nodes is the maximum). The Replication Interval can be configured to save on the connection setup overhead.

Typical machine behavior

Turning a machine off overnight

Normally an active entry will be replicated to at least one other WINS server, because there will be at least one replication partner with a relatively short Replication Interval. After some time it will have been replicated to all other WINS servers. When a machine is shutdown at the end of the day, it will release the name. With the default Extinction Interval it will not change into the extinct state during the night and therefore it will not be replicated again that night. When it is booted the next morning the machine will register the name again at the same WINS server, where it will get a new version ID. This entry is replicated just like it was replicated the previous day.

Therefore, the number of entries which are replicated more or less every working day are the number of nodes which are switched on in the morning times the number of names per node.

Roving user When a user powers down the machine and then moves to a

different subnet with another primary WINS server there will be name challenge traffic. Typically the Name Registration Request will be answered with the Wait for Acknowledgment message (100 bytes). Then the new WINS will (assuming the active entry was replicated) challenge the IP address which is currently in its database for this name (Name Query packet, 92 bytes), when there is no reply, as can be expected in this case, it will try that three times. Usually the name challenge will never travel over the subnet where the machine has left because the ARP will fail, but the challenge message can be seen on the subnet of the new WINS server and the links between the routers. The entry will get a new version ID so it will be replicated from its new owner to the other WINS servers.

A detailed network trace of this machine behavior can be found in Appendix C.

Stationary machines Some machines will not boot for a period which is longer than the *Verify Interval*. The replicas of entries of these machines will not be deleted, because the owned entry will never become extinct. They will remain active in the database. A machine which boots sometimes will cause all of its replica entries to be “refreshed” by replicas with a new version ID. Sometimes the machine will not be booted and therefore its replicas will not be “refreshed” for a period which is longer than the *Verify Interval*. When a WINS server has such an old replica it will attempt to verify this entry with the owning WINS server. Most WINS servers will not scavenge at exactly the same time, and therefore the verify traffic is not concentrated into a peak load on the network. But when a large number of WINS servers is involved then you might want to consider this load.

Joining two WINS systems When two organizations merge, the machines of these organizations may have to connect to each other. Therefore their WINS systems have to be joined. There are two things that one has to be aware of, the initial replication load and possible name conflicts. When the first two WINS servers are connected they will at some moment start replication. Since all entries are from new owners the whole database will be replicated. Then their replication partners will get the new entries and so on. You may want to merge when the connecting WAN links are more or less idle. When the databases contain conflicting names the conflicts are resolved as described in the earlier chapters, which may result in other traffic. The users of machines with conflicting names will probably call the help desk, when they suddenly get the “duplicate name” messages and their machine will refuse to open new sessions.

Estimates of the total amount of traffic We can make some estimates of the WINS traffic when the behavior of the clients can be described in terms of the above cases: booting machines, roving users or stationary machines. But it all depends on the topology of the WINS servers design and the links between them. It may not always be possible to predict the load on a specific link because the routers may decide rather autonomously how they will route the traffic.

For large networks (100,000+ nodes) the biggest load will probably be the replication load caused by the entries of the daily booting of machines. The difference in time zones may spread this load, over the day. When the replication network traffic over a wide area network becomes more than the registration and query load, it is better not to use local WINS servers.

SUMMARY

WINS is designed to make it possible to build a fault tolerant distributed NetBIOS name-to-IP address database. By keeping the number of WINS servers small, replication traffic can be limited. An estimate of the WINS traffic can be made when an estimate of the client behavior is available. The default timers, generally don't need to be modified.

In general, the idea is to keep the number of WINS servers as low as possible. Whenever in doubt use the alternative with less WINS servers. But, design in such a way that the configuration can be expanded later as the network grows or as other networks join the current configuration. The number of WINS servers should be large enough to allow for fault tolerance in the design.

When the number of WINS servers grows there has to be some hierarchy in the replication configuration. In a large network it is seldom a good idea to make all WINS servers replication partners of each other. The convergence time will increase when the replication path becomes longer.

For more information

For the latest information on Windows NT Server, check out our World Wide Web site at <http://www.microsoft.com/backoffice> or the Windows NT Server Forum on the Microsoft Network (GO WORD: MSNTS).

APPENDIX A: TERMINOLOGY AND NETBIOS NAMES

Terminology

There are various means of name-to-IP address mapping for name resolution.

Term	Definition
B-NODE	Broadcast nodes communicate using a mix of UDP datagrams (both broadcast and directed) and TCP connections. They interoperate with one another within a broadcast area but cannot interoperate across routers in a routed network. B-nodes generate high-broadcast traffic. Each node on the LAN must examine every broadcast datagram.
P-NODE	Point-to-point nodes communicate using only directed UDP datagrams and TCP sessions. They rely on NetBIOS name servers, local or remote. If the name server is down, the p-node cannot communicate with any other system even those on the same local network.
M-NODE	Mixed nodes are p-nodes which have been given certain b-node characteristics. M-nodes use broadcast first (to optimize performance, assuming that most resources reside on the local broadcast medium) for name registration and resolution. If this is unsuccessful, point-to-point communication with the name server is used. M-nodes generate high-broadcast traffic, but can cross routers and continue to operate normally if the name server is down.
H-NODE	Hybrid nodes (currently if RFC draft form) are also a combination of b-node and p-node functionality. H-node uses point-to-point communication first. If the NetBIOS name server cannot be located, it switches to broadcast. H-node continues to poll for the name server and returns to point-to-point communication when one becomes available.

NetBIOS Names

Microsoft networking components, such as Windows NT Workstation and Windows NT Server services, allow the first 15 characters of a NetBIOS name to be specified by the user or administrator, but reserve the 16th character of the NetBIOS name (00-FF hex) to indicate a resource type. Following are some examples of NetBIOS names used by Microsoft components:

Unique Names

`\\computer_name[00h]`

Registered by the Workstation Service on the WINS Client

`\\computer_name[03h]`

Registered by the Messenger Service on the WINS Client

`\\computer_name[06h]`

Registered by the Remote Access Service (RAS), when started on a RAS Server.

`\\computer_name[1Fh]`

Registered by the Network Dynamic Data Exchange (NetDDE) services, will only appear if the NetDDE services are started on the computer. By default under Windows NT 3.51, the NetDDE services are not automatically started.

`\\computer_name[20h]`

Registered by the Server Service on the WINS Client.

`\\computer_name[21h]`

Registered by the RAS Client Service, when started on a RAS Client.

`\\computer_name[BEh]`

Registered by the Network Monitoring Agent Service—will only appear if the service is started on the computer. If the computer name is not a full 15 characters, the name will be padded with plus (+) symbols.

`\\computer_name[BFh]`

Registered by the Network Monitoring Utility (included with Microsoft Systems Management Server). If the computer name is not a full 15 characters, the name will be padded with plus (+) symbols.

\\username[03h]

User names for the currently logged on users are registered in the WINS database. The username is registered by the Server component so that the user can receive any "net send" commands sent to their username. If more than one user is logged on with the same username, only the first computer at which a user logged on with the username will register the name.

\\domain_name[1Bh]

Registered by the Windows NT Server primary domain controller (PDC) that is running as the Domain Master Browser and is used to allow remote browsing of domains. When a WINS server is queried for this name, a WINS server returns the IP address of the computer that registered this name.

\\domain_name[1Dh]

Registered only by the Master Browser, of which there can only be one for each subnet. This name is used by the Backup Browsers to communicate with the Master Browser to retrieve the list of available servers from the Master Browser.

WINS servers always return a positive registration response for domain_name[1D], even though the WINS server does not "register" this name in its database. Therefore, when a WINS server is queried for the domain_name[1D], the WINS server returns a negative response, which will cause the client to broadcast to resolve the name.

Group Names

\\domain_name[00h]

Registered by the Workstation Service so that it can receive browser broadcasts from LAN Manager-based computers.

\\domain_name[1Ch]

Registered for use by the domain controllers within the domain and can contain up to 25 IP addresses. One IP address will be that of the primary domain controller (PDC) and the other 24 will be the IP addresses of backup domain controllers (BDCs).

\\domain_name[1Eh]

Registered for browsing purposes and is used by the browsers to elect a Master Browser (this is how a statically mapped group name will register itself). When a WINS server receives a name query for a name ending with [1E], the WINS server will always return the network broadcast address for the requesting client's local network.

\\--__MSBROWSE__[01h]

Registered by the Master Browser for each subnet. When a WINS server receives a name query for this name, the WINS server will always return the network broadcast address for the requesting client's local network.

APPENDIX B: NETBT (NETBIOS OVER TCP) CONFIGURATION PARAMETERS

Introduction

All of the NetBT parameters are registry values located under one of two different subkeys of HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services:

- Netbt\Parameters
- Netbt\Adapters\<Adapter Name>, in which <Adapter Name> refers the subkey for a network adapter that NetBT is bound to, such as Lance01.

Values under the latter key(s) are specific to each adapter. If the system is configured via DHCP, then a change in parameters will take effect if the command *ipconfig /renew* is issued in a command shell. Otherwise, a reboot of the system is required for a change in any of these parameters to take effect.

Standard Parameters Configurable from the Registry Editor

The following parameters are installed with default values by the NCPA during the installation of the TCP/IP components. They may be modified using the Registry Editor (regedt32.exe).

BcastNameQueryCount

Key: Netbt\Parameters

Value Type: REG_DWORD - Count

Valid Range: 1 to 0xFFFF

Default: 3

Description: This value determines the number of times NetBT broadcasts a query for a given name without receiving a response.

BcastQueryTimeout

Key: Netbt\Parameters

Value Type: REG_DWORD - Time in milliseconds

Valid Range: 100 to 0xFFFFFFFF

Default: 0x2ee (750 decimal)

Description: This value determines the time interval between successive broadcast name queries for the same name.

CacheTimeout

Key: Netbt\Parameters

Value Type: REG_DWORD - Time in milliseconds

Valid Range: 60000 to 0xFFFFFFFF

Default: 0x927c0 (600000 milliseconds = 10 minutes)

Description: This value determines the time interval that names are cached in the remote name table.

NameServerPort

Key: Netbt\Parameters

Value Type: REG_DWORD - UDP port number

Valid Range: 0 - 0xFFFF

Default: 0x89

Description: This parameter determines the destination port number to which NetBT will send name service related packets, such as name queries and name registrations, to WINS. The Microsoft WINS listens on port 0x89. NetBIOS name servers from other vendors may listen on different ports.

NameSrvQueryCount

Key: Netbt\Parameters

Value Type: REG_DWORD - Count

Valid Range: 0 - 0xFFFF

Default: 3

Description: This value determines the number of times NetBT sends a query to a WINS server for a given name without receiving a response.

NameSrvQueryTimeout

Key: Netbt\Parameters

Value Type: REG_DWORD - Time in milliseconds

Valid Range: 100 - 0xFFFFFFFF

Default: 1500 (1.5 seconds)

Description: This value determines the time interval between successive name queries to WINS for a given name.

SessionKeepAlive

Key: Netbt\Parameters

Value Type: REG_DWORD - Time in milliseconds

Valid Range: 60,000 - 0xFFFFFFFF

Default: 3,600,000 (1 hour)

Description: This value determines the time interval between keepalive transmissions on a session. Setting the value to 0xFFFFFFFF disables keepalives.

Size/Small/Medium/Large

Key: Netbt\Parameters

Value Type: REG_DWORD

Valid Range: 1, 2, 3 (Small, Medium, Large)

Default: 1 (Small)

Description: This value determines the size of the name tables used to store local and remote names. In general, Small is adequate. If the system is acting as a proxy nameserver, then the value is automatically set to Large to increase the size of the name cache hash table. Hash table buckets are sized as follows:

Large: 256 Medium: 128 Small: 16

Optional Parameters Configurable from the Registry Editor

These parameters normally do not exist in the registry. They may be created to modify the default behavior of the NetBT protocol driver.

BroadcastAddress

Key: Netbt\Parameters

Value Type: REG_DWORD - Four byte, little-endian encoded IP address

Valid Range: 0 - 0xFFFFFFFF

Default: The ones-broadcast address for each network.

Description: This parameter can be used to force NetBT to use a specific address for all broadcast name related packets. By default, NetBT uses the ones-broadcast address appropriate for each net (that is for a network of 11.101.0.0 with a subnet mask of 255.255.0.0, the subnet broadcast address would be 11.101.255.255). This parameter would be set, for example, if the network uses the zeros-broadcast address (set using the UseZeroBroadcast TCP/IP parameter). The appropriate subnet broadcast address would then be 11.101.0.0 in the example above. This parameter would then be set to 0x0b650000. Note that this parameter is global and will be used on all subnets that NetBT is bound to.

EnableProxyRegCheck

Key: Netbt\Parameters

Value Type: REG_DWORD - Boolean

Valid Range: 0 or 1 (False or True)

Default: 0 (False)

Description: If this parameter is set to 1 (True), then the proxy name server will send a negative response to a broadcast name registration if the name is already registered with WINS or is in the proxy's local name cache with a different IP address. The hazard of enabling this feature is that it prevents a system from changing its IP address as long as WINS has a mapping for the name. For this reason, it is disabled by default.

InitialRefreshTimeout

Key: Netbt\Parameters

Value Type: REG_DWORD - Time in milliseconds

Valid Range: 960000 - 0xFFFFFFFF

Default: 960000 (16 minutes)

Description: This parameter specifies the initial refresh timeout used by NetBT during name registration. NetBT tries to contact the WINS servers at 1/8th of this time interval when it is first registering names. When it receives a successful registration response, that response will contain the new refresh interval to use.

LmhostsTimeout

Key: Netbt\Parameters

Value Type: REG_DWORD - Time in milliseconds

Valid Range: 1000 - 0xFFFFFFFF

Default: 6000 (6 seconds)

Description: This parameter specifies the timeout value for Lmhosts and DNS name queries. The timer has a granularity of the timeout value, so the actual timeout could be as much as twice the value.

MaxDgramBuffering

Key: Netbt\Parameters

Value Type: REG_DWORD - Count of bytes

Valid Range: 0 - 0xFFFFFFFF

Default: 0x20000 (128 Kb)

Description: This parameter specifies the maximum amount of memory that NetBT will dynamically allocate for all outstanding datagram sends. Once this limit is reached, further sends will fail due to insufficient resources.

NodeType

Key: Netbt\Parameters

Value Type: REG_DWORD - Number

Valid Range: 1,2,4,8 (B-node, P-node, M-node, H-node)

Default: 1 or 8 based on the WINS server configuration

Description: This parameter determines what methods NetBT will use to register and resolve names. A B-node system uses broadcasts. A P-node system uses only point-to-point name queries to a name server (WINS). An M-node system uses broadcasts first, then queries the name server. An H-node system queries the name server first, then broadcasts. Resolution via LMHOSTS and/or DNS, if enabled, will follow these methods. If this key is present it will override the DhcpNodeType key. If neither key is present, the system defaults to B-node if there are no WINS servers configured for the network. The system defaults to H-node if there is at least one WINS server configured.

RandomAdapter

Key: Netbt\Parameters

Value Type: REG_DWORD - Boolean

Valid Range: 0 or 1 (False or True)

Default: 0 (False)

Description: This parameter applies to a multihomed host only. If it is set to 1 (True), then NetBT will randomly choose the IP address to put in a name query response from all of its bound interfaces. Usually, the response contains the address of the interface that the query arrived on. This feature would be used by a server with two interfaces on the same network for load balancing.

RefreshOpCode

Key: Netbt\Parameters

Value Type: REG_DWORD - Number

Valid Range: 8, 9

Default: 8

Description: This parameter forces NetBT to use a specific opcode in name refresh packets. The specification for the NetBT protocol is somewhat ambiguous

in this area. Although the default of 8 used by Microsoft implementations appears to be the intended value, some other implementations, such as those by Ungermann-Bass, use the value 9. Two implementations must use the same opcode to interoperate.

SingleResponse

Key: Netbt\Parameters

Value Type: REG_DWORD - Boolean

Valid Range: 0 or 1 (False or True)

Default: 0 (False)

Description: This parameter applies to a multihomed host only. If this parameter is set to 1 (True), then NetBT will only supply an IP address from one of its bound interfaces in name query responses. By default, the addresses of all bound interfaces are included.

WinsDownTimeout

Key: Netbt\Parameters

Value Type: REG_DWORD - Time in milliseconds

Valid Range: 1000 - 0xFFFFFFFF

Default: 15,000 (15 seconds)

Description: This parameter determines the amount of time NetBT will wait before again trying to use WINS after it fails to contact any WINS server. This feature primarily allows computers that are temporarily disconnected from the network, such as laptops, to proceed through boot processing without waiting to timeout out each WINS name registration or query individually.

Parameters Configurable from the Network Control Panel Applet

The following parameters can be set via the NCPA. There should be no need to configure them directly.

EnableDns

Key: Netbt\Parameters

Value Type: REG_DWORD - Boolean

Valid Range: 0 or 1 (False or True)

Default: 0 (False)

Description: If this value is set to 1 (True), then NetBT will query the DNS for names that cannot be resolved by WINS, broadcast, or the LMHOSTS file.

EnableLmhosts

Key: Netbt\Parameters

Value Type: REG_DWORD - Boolean

Valid Range: 0 or 1 (False or True)

Default: 1 (True)

Description: If this value is set to 1 (True), then NetBT will search the LMHOSTS file, if it exists, for names that cannot be resolved by WINS or broadcast. By default there is no Lmhosts file database directory (specified by Tcpip\Parameters\DatabasePath), so no action will be taken. This value is written by the Advanced TCP/IP configuration dialog of the NCPA.

EnableProxy

Key: Netbt\Parameters

Value Type: REG_DWORD - Boolean

Valid Range: 0 or 1 (False or True)

Default: 0 (False)

Description: If this value is set to 1 (True), then the system will act as a proxy name server for the networks to which NetBT is bound. A proxy name server answers broadcast queries for names that it has resolved through WINS. A proxy nameserver allows a network of B-node implementations to connect to servers on other subnets that are registered with WINS.

NameServer

Key: Netbt\Adapters\<>Adapter Name>

Value Type: REG_SZ - Dotted decimal IP address (i.e. 11.101.1.200)

Valid Range: Any valid IP address

Default: blank (no address)

Description: This parameter specifies the IP address of the primary WINS server. If this parameter contains a valid value, it overrides the DHCP parameter of the same name.

NameServerBackup

Key: Netbt\Adapters\<>Adapter Name>

Value Type: REG_SZ - Dotted decimal IP address (i.e. 11.101.1.200)

Valid Range: Any valid IP address.

Default: blank (no address)

Description: This parameter specifies the IP address of the backup WINS server. If this parameter contains a valid value, it overrides the DHCP parameter of the same name.

Scopeld

Key: Netbt\Parameters

Value Type: REG_SZ - Character string

Valid Range: Any valid DNS domain name consisting of two dot-separated parts, or a "*".

Default: None

Description: This parameter specifies the NetBIOS name scope for the node. This value must not begin with a period. If this parameter contains a valid value, it will override the DHCP parameter of the same name. A blank value (empty string) will be ignored. Setting this parameter to the value "*" indicates a null scope and will override the DHCP parameter.

Non-Configurable Parameters

The following parameters are created and used internally by the NetBT components. They should never be modified using the Registry Editor. They are listed here for reference only.

DhcpNameServer

Key: Netbt\Adapters\<>Adapter Name>

Value Type: REG_SZ - Dotted decimal IP address (i.e. 11.101.1.200)

Valid Range: Any valid IP address

Default: None

Description: This parameter specifies the IP address of the primary WINS server. It is written by the DHCP client service, if enabled. A valid NameServer value will override this parameter.

DhcpNameServerBackup

Key: Netbt\Adapters\<>Adapter Name>

Value Type: REG_SZ - Dotted decimal IP address (i.e. 11.101.1.200)

Valid Range: Any valid IP address

Default: None

Description: This parameter specifies the IP address of the backup WINS server. It is written by the DHCP client service, if enabled. A valid BackupNameServer value will override this parameter.

DhcpNodeType

Key: Netbt\Parameters

Value Type: REG_DWORD - Number

Valid Range: 1 - 8

Default: 1

Description: This parameter specifies the NetBT node type. It is written by the DHCP client service, if enabled. A valid NodeType value will override this parameter. See the entry for NodeType for a complete description.

DhcpScopeld

Key: Netbt\Parameters

Value Type: REG_SZ - Character string

Valid Range: a dot separated name string such as "microsoft.com"

Default: None

Description: This parameter specifies the NetBIOS name scope for the node. It is written by the DHCP client service, if enabled. This value must **not** begin with a period. See the entry for Scopeld for more information.

NbProvider

Key: Netbt\Parameters

Value Type: REG_SZ - Character string

Valid Range: _tcp

Default: _tcp

Description: This parameter is used internally by the RPC component. The default value should not be changed.

TransportBindName

Key: Netbt\Parameters

Value Type: REG_SZ - Character string

Valid Range: N/A

Default: \Device\

Description: This parameter is used internally during product development. The default value should not be changed.

**APPENDIX C:
NAME REGISTRATION
TRACE**

Trace of a WINS client which was forced to use a new IP address (same subnet) and register with a different primary WINS server (on a different subnet)

Part 1 Name Registration Request, WACK and Registration Response

The name registration request of the new IP address.

```
- - - - - Frame 642 - - - - -
Frame Time Src MAC Addr Dst MAC Addr Protocol Description      Src Other Addr Dst Other Addr Type Other Addr
642 1115.71 DEL486NT MCDNT  NBT NS: MultiHomed Name Registration req. for MC New Client IP IP MCDNT IP

+ FRAME: Base frame properties
+ TOKENRING: Length = 118, Priority Normal (No token) LLC Frame
+ LLC: UI DSAP=0xAA SSAP=0xAA C
+ SNAP: ETYPE = 0x0800
  IP: ID = 0x200; Proto = UDP; Len: 96
  IP: Version = 4 (0x4)
  IP: Header Length = 20 (0x14)
  + IP: Service Type = 0 (0x0)
  IP: Total Length = 96 (0x60)
  IP: Identification = 512 (0x200)
  + IP: Flags Summary = 0 (0x0)
  IP: Fragment Offset = 0 (0x0) bytes
  IP: Time to Live = 31 (0x1F)
  IP: Protocol = UDP - User Datagram
  IP: CheckSum = 0x8576
  IP: Source Address = 10.0.0.18
  IP: Destination Address = 10.0.0.6
  IP: Data: Number of data bytes remaining = 76 (0x004C)
  UDP: Src Port: NETBIOS Name Service, (137); Dst Port: NETBIOS Name Service (137); Length = 76 (0x4C)
  UDP: Source Port = NETBIOS Name Service
  UDP: Destination Port = NETBIOS Name Service
  UDP: Total length = 76 (0x4C) bytes
  UDP: CheckSum = 0xB993
  UDP: Data: Number of data bytes remaining = 68 (0x0044)
  NBT: NS: MultiHomed Name Registration req. for MCSPAULLEM2 <00>
  NBT: Transaction ID = 32768 (0x8000)
  NBT: Flags Summary = 0x7900 - Req.; MultiHomed Name Registration; Success
  NBT: 0..... = Request
  NBT: .1111..... = MultiHomed Name Registration
  NBT: .....0..... = Non-authoritative Answer
  NBT: .....0..... = Datagram not truncated
  NBT: .....1..... = Recursion desired
  NBT: .....0..... = Recursion not available
  NBT: .....0..... = Reserved
  NBT: .....0..... = Reserved
  NBT: .....0.... = Not a broadcast packet
  NBT: .....0000 = Success
```

```

NBT: Question Count = 1 (0x1)
NBT: Answer Count = 0 (0x0)
NBT: Name Service Count = 0 (0x0)
NBT: Additional Record Count = 1 (0x1)
NBT: Question Name = MCSPAULLEM2 <00>
NBT: Question Type = General Name Service
NBT: Question Class = Internet Class
NBT: Resource Record Name = MCSPAULLEM2 <00>
NBT: Resource Record Type = NetBIOS General Name Service
NBT: Resource Record Class = Internet Class
NBT: Time To Live = 300000 (0x493E0)
NBT: RDATA Length = 6 (0x6)
+ NBT: Resource Record Flags = 24576 (0x6000)
NBT: Owner IP Address = 10.0.0.18

```

```

00000: 10 40 10 00 5A 3B 43 7F 10 00 5A 78 12 6C AA AA .@..Z;C...Zx.l..
00010: 03 00 00 00 08 00 45 00 00 60 02 00 00 00 1F 11 .....E..`.....
00020: 85 76 0A 00 00 12 0A 00 00 06 00 89 00 89 00 4C .v.....L
00030: B9 93 80 00 79 00 00 01 00 00 00 00 00 01 20 45 ....y..... E
00040: 4E 45 44 46 44 46 41 45 42 46 46 45 4D 45 4D 45 NEDFDFAEBFFEMEME
00050: 46 45 4E 44 43 43 41 43 41 43 41 43 41 41 41 00 FENDCCACACACAAA.
00060: 00 20 00 01 C0 0C 00 20 00 01 00 04 93 E0 00 06 . .....
00070: 60 00 0A 00 00 12      `.....

```

The client will have to wait: send WACK.

```

- - - - - Frame 643 - - - - -
Frame Time Src MAC Addr Dst MAC Addr Protocol Description      Src Other Addr Dst Other Addr Type Other Addr
643 1115.72 MCDNT DEL486NT NBT NS: WACK (Node Status) resp. for MCSPAULLEM2 IP MCDNT New Client IP IP

```

```

+ FRAME: Base frame properties
+ TOKENRING: Length = 108, Priority Normal (No token) LLC Frame
+ LLC: UI DSAP=0xAA SSAP=0xAA C
+ SNAP: ETYPE = 0x0800
IP: ID = 0x906; Proto = UDP; Len: 86
IP: Version = 4 (0x4)
IP: Header Length = 20 (0x14)
+ IP: Service Type = 0 (0x0)
IP: Total Length = 86 (0x56)
IP: Identification = 2310 (0x906)
+ IP: Flags Summary = 0 (0x0)
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 32 (0x20)
IP: Protocol = UDP - User Datagram
IP: CheckSum = 0x7D7A

```

```

IP: Source Address = 10.0.0.6
IP: Destination Address = 10.0.0.18
IP: Data: Number of data bytes remaining = 66 (0x0042)
UDP: Src Port: NETBIOS Name Service, (137); Dst Port: NETBIOS Name Service (137); Length = 66 (0x42)
UDP: Source Port = NETBIOS Name Service
UDP: Destination Port = NETBIOS Name Service
UDP: Total length = 66 (0x42) bytes
UDP: CheckSum = 0xBBCE
UDP: Data: Number of data bytes remaining = 58 (0x003A)
NBT: NS: WACK (Node Status) resp. for MCSPAULLEM2 <00>, Success
NBT: Transaction ID = 32768 (0x8000)
NBT: Flags Summary = 0xBC00 - Resp.; WACK; Success
NBT: 1..... = Response
NBT: .0111..... = WACK
NBT: .....1..... = Authoritative Answer
NBT: .....0..... = Datagram not truncated
NBT: .....0..... = Recursion not desired
NBT: .....0..... = Recursion not available
NBT: .....0..... = Reserved
NBT: .....0..... = Reserved
NBT: .....0.... = Not a broadcast packet
NBT: .....0000 = Success
NBT: Question Count = 0 (0x0)
NBT: Answer Count = 1 (0x1)
NBT: Name Service Count = 0 (0x0)
NBT: Additional Record Count = 0 (0x0)
NBT: Resource Record Name = MCSPAULLEM2 <00>
NBT: Resource Record Type = NetBIOS General Name Service
NBT: Resource Record Class = Internet Class
NBT: Time To Live = 2 (0x2)
NBT: RDATA Length = 2 (0x2)
NBT: Flags Summary = 0x7900 - Req.; MultiHomed Name Registration; Success
NBT: 0..... = Request
NBT: .0000..... = Query
NBT: .....0..... = Non-authoritative Answer
NBT: .....0..... = Datagram not truncated
NBT: .....1..... = Recursion desired
NBT: .....0..... = Recursion not available
NBT: .....0..... = Reserved
NBT: .....0..... = Reserved
NBT: .....0.... = Not a broadcast packet

00000: 10 40 10 00 5A 78 12 6C 10 00 5A 3B 43 7F AA AA .@..Zx.l..Z;C[]..
00010: 03 00 00 00 08 00 45 00 00 56 09 06 00 00 20 11 .....E..V....
00020: 7D 7A 0A 00 00 06 0A 00 00 12 00 89 00 89 00 42 }z.....B

```

```

00030: BB CE 80 00 BC 00 00 00 01 00 00 00 20 45 ..... E
00040: 4E 45 44 46 44 46 41 45 42 46 46 45 4D 45 4D 45 NEDFDFAEBFFEMEME
00050: 46 45 4E 44 43 43 41 43 41 43 41 43 41 41 41 00 FENDCCACACACAAA.
00060: 00 20 00 01 00 00 02 00 02 79 00 . .....y.

```

The challenge of the old IP address of the client (actually there are three, almost identical, frames like this but only one is printed)

```

- - - - - Frame 644 - - - - -
Frame Time Src MAC Addr Dst MAC Addr Protocol Description Src Other Addr Dst Other Addr Type Other Addr
644 1115.78 MCDNT DEL486NT NBT NS: Query req. for MCSPAULLEM2 <00> IP MCDNT Old Client IP IP

```

```

+ FRAME: Base frame properties
+ TOKENRING: Length = 100, Priority Normal (No token) LLC Frame
+ LLC: UI DSAP=0xAA SSAP=0xAA C
+ SNAP: ETYPE = 0x0800
IP: ID = 0xA06; Proto = UDP; Len: 78
IP: Version = 4 (0x4)
IP: Header Length = 20 (0x14)
+ IP: Service Type = 0 (0x0)
IP: Total Length = 78 (0x4E)
IP: Identification = 2566 (0xA06)
+ IP: Flags Summary = 0 (0x0)
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 32 (0x20)
IP: Protocol = UDP - User Datagram
IP: CheckSum = 0x7C81
IP: Source Address = 10.0.0.6
IP: Destination Address = 10.0.0.19
IP: Data: Number of data bytes remaining = 58 (0x003A)
UDP: Src Port: NETBIOS Name Service, (137); Dst Port: NETBIOS Name Service (137); Length = 58 (0x3A)
UDP: Source Port = NETBIOS Name Service
UDP: Destination Port = NETBIOS Name Service
UDP: Total length = 58 (0x3A) bytes
UDP: CheckSum = 0x6FE3
UDP: Data: Number of data bytes remaining = 50 (0x0032)
NBT: NS: Query req. for MCSPAULLEM2 <00>
NBT: Transaction ID = 0 (0x0)
NBT: Flags Summary = 0x0100 - Req.; Query; Success
NBT: 0..... = Request
NBT: .0000..... = Query
NBT: .....0..... = Non-authoritative Answer
NBT: .....0..... = Datagram not truncated
NBT: .....1..... = Recursion desired
NBT: .....0..... = Recursion not available

```

```

NBT: .....0..... = Reserved
NBT: .....0..... = Reserved
NBT: .....0.... = Not a broadcast packet
NBT: .....0000 = Success
NBT: Question Count = 1 (0x1)
NBT: Answer Count = 0 (0x0)
NBT: Name Service Count = 0 (0x0)
NBT: Additional Record Count = 0 (0x0)
NBT: Question Name = MCSPAULLEM2 <00>
NBT: Question Type = General Name Service
NBT: Question Class = Internet Class

```

```

00000: 10 40 10 00 5A 78 12 6C 10 00 5A 3B 43 7F AA AA .@..Zx.l..Z;C[]..
00010: 03 00 00 00 08 00 45 00 00 4E 0A 06 00 00 20 11 .....E..N....
00020: 7C 81 0A 00 00 06 0A 00 00 13 00 89 00 89 00 3A |.....:
00030: 6F E3 00 00 01 00 00 01 00 00 00 00 00 00 20 45 o..... E
00040: 4E 45 44 46 44 46 41 45 42 46 46 45 4D 45 4D 45 NEDFDFAEBFFEMEME
00050: 46 45 4E 44 43 43 41 43 41 43 41 43 41 41 41 00 FENDCCACACACAAA.
00060: 00 20 00 01 . . .

```

The positive name registration response.

- - - - - Frame 647 - - - - -

Frame	Time	Src MAC	Addr	Dst MAC	Addr	Protocol	Description	Src Other	Addr	Dst Other	Addr	Type	Other	Addr
647	1117.32	MCDNT	DEL486NT	NBT	NS: Registration (Node Status)	resp. for MCS	IP MCDNT	New Client	IP	IP				

```

+ FRAME: Base frame properties
+ TOKENRING: Length = 112, Priority Normal (No token) LLC Frame
+ LLC: UI DSAP=0xAA SSAP=0xAA C
+ SNAP: ETYPE = 0x0800
IP: ID = 0xD06; Proto = UDP; Len: 90
IP: Version = 4 (0x4)
IP: Header Length = 20 (0x14)
+ IP: Service Type = 0 (0x0)
IP: Total Length = 90 (0x5A)
IP: Identification = 3334 (0xD06)
+ IP: Flags Summary = 0 (0x0)
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 32 (0x20)
IP: Protocol = UDP - User Datagram
IP: CheckSum = 0x7976
IP: Source Address = 10.0.0.6
IP: Destination Address = 10.0.0.18
IP: Data: Number of data bytes remaining = 70 (0x0046)
UDP: Src Port: NETBIOS Name Service, (137); Dst Port: NETBIOS Name Service (137); Length = 70 (0x46)

```

```
UDP: Source Port = NETBIOS Name Service
UDP: Destination Port = NETBIOS Name Service
UDP: Total length = 70 (0x46) bytes
UDP: CheckSum = 0xCB22
UDP: Data: Number of data bytes remaining = 62 (0x003E)
NBT: NS: Registration (Node Status) resp. for MCSPAULLEM2 <00>, Success, Owner Addr. 10.0.0.18
NBT: Transaction ID = 32768 (0x8000)
NBT: Flags Summary = 0xAD80 - Resp.; Registration; Success
  NBT: 1..... = Response
  NBT: .0101..... = Registration
  NBT: .....1..... = Authoritative Answer
  NBT: .....0..... = Datagram not truncated
  NBT: .....1..... = Recursion desired
  NBT: .....1..... = Recursion available
  NBT: .....0..... = Reserved
  NBT: .....0..... = Reserved
  NBT: .....0.... = Not a broadcast packet
  NBT: .....0000 = Success
NBT: Question Count = 0 (0x0)
NBT: Answer Count = 1 (0x1)
NBT: Name Service Count = 0 (0x0)
NBT: Additional Record Count = 0 (0x0)
NBT: Resource Record Name = MCSPAULLEM2 <00>
NBT: Resource Record Type = NetBIOS General Name Service
NBT: Resource Record Class = Internet Class
NBT: Time To Live = 3600 (0xE10)
NBT: RDATA Length = 6 (0x6)
+ NBT: Resource Record Flags = 24576 (0x6000)
NBT: Owner IP Address = 10.0.0.18
```

```
00000: 10 40 10 00 5A 78 12 6C 10 00 5A 3B 43 7F AA AA .@..Zx.l..Z;C[]..
00010: 03 00 00 00 08 00 45 00 00 5A 0D 06 00 00 20 11 .....E..Z....
00020: 79 76 0A 00 00 06 0A 00 00 12 00 89 00 89 00 46 yv.....F
00030: CB 22 80 00 AD 80 00 00 00 01 00 00 00 00 20 45 ."..... E
00040: 4E 45 44 46 44 46 41 45 42 46 46 45 4D 45 4D 45 NEDFDFAEBFFEMEME
00050: 46 45 4E 44 43 43 41 43 41 43 41 43 41 41 41 00 FENDCCACACACAAA.
00060: 00 20 00 01 00 00 0E 10 00 06 60 00 0A 00 00 12 . .....`.....
```

Part 2 Name Refresh Request and Response

The name refresh request.

```
- - - - - Frame 762 - - - - -
Frame Time Src MAC Addr Dst MAC Addr Protocol Description      Src Other Addr Dst Other Addr Type Other Addr
762 1231.12 DEL486NT MCDNT  NBT NS: Refresh req. for MCSPAULLEM2 <00> New Client IP IP MCDNT IP

+ FRAME: Base frame properties
+ TOKENRING: Length = 118, Priority Normal (No token) LLC Frame
+ LLC: UI DSAP=0xAA SSAP=0xAA C
+ SNAP: ETYPE = 0x0800
  IP: ID = 0x5D00; Proto = UDP; Len: 96
  IP: Version = 4 (0x4)
  IP: Header Length = 20 (0x14)
  + IP: Service Type = 0 (0x0)
  IP: Total Length = 96 (0x60)
  IP: Identification = 23808 (0x5D00)
  + IP: Flags Summary = 0 (0x0)
  IP: Fragment Offset = 0 (0x0) bytes
  IP: Time to Live = 31 (0x1F)
  IP: Protocol = UDP - User Datagram
  IP: CheckSum = 0x2A76
  IP: Source Address = 10.0.0.18
  IP: Destination Address = 10.0.0.6
  IP: Data: Number of data bytes remaining = 76 (0x004C)
  UDP: Src Port: NETBIOS Name Service, (137); Dst Port: NETBIOS Name Service (137); Length = 76 (0x4C)
  UDP: Source Port = NETBIOS Name Service
  UDP: Destination Port = NETBIOS Name Service
  UDP: Total length = 76 (0x4C) bytes
  UDP: CheckSum = 0xF25E
  UDP: Data: Number of data bytes remaining = 68 (0x0044)
  NBT: NS: Refresh req. for MCSPAULLEM2 <00>
  NBT: Transaction ID = 32821 (0x8035)
  NBT: Flags Summary = 0x4000 - Req.; Refresh; Success
  NBT: 0..... = Request
  NBT: .1000..... = Refresh
  NBT: .....0..... = Non-authoritative Answer
  NBT: .....0..... = Datagram not truncated
  NBT: .....0..... = Recursion not desired
  NBT: .....0..... = Recursion not available
  NBT: .....0..... = Reserved
  NBT: .....0..... = Reserved
  NBT: .....0.... = Not a broadcast packet
  NBT: .....0000 = Success
```

```

NBT: Question Count = 1 (0x1)
NBT: Answer Count = 0 (0x0)
NBT: Name Service Count = 0 (0x0)
NBT: Additional Record Count = 1 (0x1)
NBT: Question Name = MCSPAULLEM2 <00>
NBT: Question Type = General Name Service
NBT: Question Class = Internet Class
NBT: Resource Record Name = MCSPAULLEM2 <00>
NBT: Resource Record Type = NetBIOS General Name Service
NBT: Resource Record Class = Internet Class
NBT: Time To Live = 300000 (0x493E0)
NBT: RDATA Length = 6 (0x6)
+ NBT: Resource Record Flags = 24576 (0x6000)
NBT: Owner IP Address = 10.0.0.18

```

```

00000: 10 40 10 00 5A 3B 43 7F 10 00 5A 78 12 6C AA AA .@..Z;C[]..Zx.l..
00010: 03 00 00 00 08 00 45 00 00 60 5D 00 00 00 1F 11 .....E..`].....
00020: 2A 76 0A 00 00 12 0A 00 00 06 00 89 00 89 00 4C *v.....L
00030: F2 5E 80 35 40 00 00 01 00 00 00 00 00 01 20 45 .^.5@..... E
00040: 4E 45 44 46 44 46 41 45 42 46 46 45 4D 45 4D 45 NEDFDFAEBFFEMEME
00050: 46 45 4E 44 43 43 41 43 41 43 41 43 41 41 41 00 FENDCCACACACAAA.
00060: 00 20 00 01 C0 0C 00 20 00 01 00 04 93 E0 00 06 . .....
00070: 60 00 0A 00 00 12      `.....

```

The positive response to name refresh request.

```

- - - - - Frame 673 - - - - -
Frame Time Src MAC Addr Dst MAC Addr Protocol Description      Src Other Addr Dst Other Addr Type Other Addr
763 1231.15 MCDNT DEL486NT NBT NS: Registration (Node Status) resp. for MCS IP MCDNT New Client IP IP

```

```

+ FRAME: Base frame properties
+ TOKENRING: Length = 112, Priority Normal (No token) LLC Frame
+ LLC: UI DSAP=0xAA SSAP=0xAA C
+ SNAP: ETYPE = 0x0800
IP: ID = 0x6506; Proto = UDP; Len: 90
IP: Version = 4 (0x4)
IP: Header Length = 20 (0x14)
+ IP: Service Type = 0 (0x0)
IP: Total Length = 90 (0x5A)
IP: Identification = 25862 (0x6506)
+ IP: Flags Summary = 0 (0x0)
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 32 (0x20)
IP: Protocol = UDP - User Datagram
IP: CheckSum = 0x2176

```

```

IP: Source Address = 10.0.0.6
IP: Destination Address = 10.0.0.18
IP: Data: Number of data bytes remaining = 70 (0x0046)
UDP: Src Port: NETBIOS Name Service, (137); Dst Port: NETBIOS Name Service (137); Length = 70 (0x46)
UDP: Source Port = NETBIOS Name Service
UDP: Destination Port = NETBIOS Name Service
UDP: Total length = 70 (0x46) bytes
UDP: CheckSum = 0xCAED
UDP: Data: Number of data bytes remaining = 62 (0x003E)
NBT: NS: Registration (Node Status) resp. for MCSPAULLEM2 <00>, Success, Owner Addr. 10.0.0.18
NBT: Transaction ID = 32821 (0x8035)
NBT: Flags Summary = 0xAD80 - Resp.; Registration; Success
NBT: 1..... = Response
NBT: .0101..... = Registration
NBT: .....1..... = Authoritative Answer
NBT: .....0..... = Datagram not truncated
NBT: .....1..... = Recursion desired
NBT: .....1..... = Recursion available
NBT: .....0..... = Reserved
NBT: .....0..... = Reserved
NBT: .....0.... = Not a broadcast packet
NBT: .....0000 = Success
NBT: Question Count = 0 (0x0)
NBT: Answer Count = 1 (0x1)
NBT: Name Service Count = 0 (0x0)
NBT: Additional Record Count = 0 (0x0)
NBT: Resource Record Name = MCSPAULLEM2 <00>
NBT: Resource Record Type = NetBIOS General Name Service
NBT: Resource Record Class = Internet Class
NBT: Time To Live = 3600 (0xE10)
NBT: RDATA Length = 6 (0x6)
+ NBT: Resource Record Flags = 24576 (0x6000)
NBT: Owner IP Address = 10.0.0.18

```

```

00000: 10 40 10 00 5A 78 12 6C 10 00 5A 3B 43 7F AA AA .@..Zx.l..Z;C[]..
00010: 03 00 00 00 08 00 45 00 00 5A 65 06 00 00 20 11 .....E..Ze... .
00020: 21 76 0A 00 00 06 0A 00 00 12 00 89 00 89 00 46 !v.....F
00030: CA ED 80 35 AD 80 00 00 00 01 00 00 00 00 20 45 ...5..... E
00040: 4E 45 44 46 44 46 41 45 42 46 46 45 4D 45 4D 45 NEDFDFAEBFFEMEME
00050: 46 45 4E 44 43 43 41 43 41 43 41 43 41 41 41 00 FENDCCACACACAAA.
00060: 00 20 00 01 00 00 0E 10 00 06 60 00 0A 00 00 12 . .....`.....

```