[MS-NBTE]:
NetBIOS over TCP (NetBT) Extensions

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
05/22/2009	0.1	Major	First Release.
07/02/2009	0.1.1	Editorial	Revised and edited the technical content.
08/14/2009	1.0	Major	Updated and revised the technical content.
09/25/2009	1.0.1	Editorial	Revised and edited the technical content.
11/06/2009	1.0.2	Editorial	Revised and edited the technical content.
12/18/2009	1.0.3	Editorial	Revised and edited the technical content.
01/29/2010	2.0	Major	Updated and revised the technical content.
03/12/2010	3.0	Major	Updated and revised the technical content.
04/23/2010	4.0	Major	Updated and revised the technical content.
06/04/2010	4.0.1	Editorial	Revised and edited the technical content.
07/16/2010	5.0	Major	Significantly changed the technical content.
08/27/2010	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	6.0	Major	Significantly changed the technical content.
11/19/2010	6.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	7.0	Major	Significantly changed the technical content.
02/11/2011	7.0	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction 5
1.1 Glossary 5
1.2 References 5
1.2.1 Normative References 5
1.2.2 Informative References 6
1.3 Overview 6
1.4 Relationship to Other Protocols 6
1.5 Prerequisites/Preconditions 6
1.6 Applicability Statement 6
1.7 Versioning and Capability Negotiation 7
1.8 Vendor-Extensible Fields 7
1.9 Standards Assignments 7

2 Messages 8
2.1 Transport 8
2.2 Message Syntax 8
2.2.1 NetBIOS Name Syntax 8
2.2.2 MULTIHOMED NAME REGISTRATION REQUEST 8

3 Protocol Details 9
3.1 NetBIOS End Node Details 9
3.1.1 Abstract Data Model 9
3.1.2 Timers 9
3.1.3 Initialization 10
3.1.4 Higher-Layer Triggered Events 10
3.1.4.1 Registering a NetBIOS Name 10
3.1.4.2 Resolving a NetBIOS Name 11
3.1.4.2.1 NetBIOS Name Server Selection 11
3.1.5 Message Processing Events and Sequencing Rules 11
3.1.5.1 Handling a NAME REGISTRATION REQUEST 11
3.1.6 Timer Events 12
3.1.7 Other Local Events 12
3.1.8 Using the LMHOSTS File to Resolve a Name Query 12
3.2 NetBIOS Name Server Details 15
3.2.1 Abstract Data Model 15
3.2.2 Timers 15
3.2.3 Initialization 15
3.2.4 Higher-Layer Triggered Events 15
3.2.5 Message Processing Events and Sequencing Rules 15
3.2.5.1 Handling a NAME REGISTRATION REQUEST (GROUP) 15
3.2.5.2 Handling a MULTIHOMED NAME REGISTRATION REQUEST (GROUP) 16
3.2.5.3 Handling a MULTIHOMED NAME REGISTRATION REQUEST (UNIQUE) 16
3.2.6 Timer Events 16
3.2.7 Other Local Events 16

4 Protocol Examples 17

5 Security Considerations 19
5.1 Security Considerations for Implementers 19
5.2 Index of Security Parameters 19

6 Appendix A: Product Behavior 20

7 Change Tracking 22

8 Index 23

1 Introduction

This document specifies extensions to the NetBIOS over TCP (NetBT) protocol, as specified in [RFC1001] and [RFC1002]. These extensions modify the syntax of allowable NetBIOS names and the behavior of timers, and add support for multihomed hosts.

1.1 Glossary

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

code page
Internet host name
NetBIOS name
NetBIOS Name Server (NBNS)
NetBIOS over TCP/IP (NetBT)
Windows Internet Name Service (WINS)
Universal Naming Convention (UNC)

The following terms are specific to this document:

group name: As defined in [RFC1002], a NetBIOS name that can be owned by any number of nodes.

LMHOST: A text file that contains entries that individually map a computer name or a NetBIOS service name to an IPv4 address. The LMHOST file is consulted when normal NetBIOS name resolution protocols fail on the wire. This legacy file is no longer installed by default on Windows systems. LM stands for "LAN Manager".

multihomed: Having two or more network interfaces on which NetBIOS over TCP is enabled.

unique name: As defined in [RFC1002], a NetBIOS name that can be owned by a single node.

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

1.2 References

1.2.1 Normative References

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

[HYBRID] Noon, F., "HYBRID NETBIOS END-NODES", April 1993, http://tools.ietf.org/id/draft-noon-hybrid-netbios-01.txt

[RFC1001] Network Working Group, "Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods", STD 19, RFC 1001, March 1987, http://www.ietf.org/rfc/rfc1001.txt

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

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

[RFC2132] Alexander, S., and Droms, R., "DHCP Options and BOOTP Vendor Extensions", RFC 2132, March 1997, http://www.ietf.org/rfc/rfc2132.txt

1.2.2 Informative References

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

[RFC1035] Mockapetris, P., "Domain Names - Implementation and Specification", STD 13, RFC 1035, November 1987, http://www.ietf.org/rfc/rfc1035.txt

[RFC4795] Aboba, B., Thaler, D., and Esibov, L., "Link-Local Multicast Name Resolution (LLMNR)", RFC 4795, January 2007, http://www.ietf.org/rfc/rfc4795.txt

1.3 Overview

NetBIOS resources are referenced by **NetBIOS name**. An application, representing a resource, registers one or more NetBIOS names that it uses to communicate with other hosts on the network. This document does the following:

1. It discusses NetBIOS name registration and name querying on hosts with more than one interface.

2. When a name server receives a registration for a **group name**, [RFC1002] requires that the name server replace any existing entry with the new entry, so that only one IP address can be registered for a group name. However, a group name is one that can be owned by any number of nodes. This document modifies the behavior of group name registrations to allow the name server to keep multiple addresses.

1.4 Relationship to Other Protocols

A NetBIOS name may or may not be derived from an **Internet host name**. The syntax for an Internet host name is much more constrained than the syntax for an arbitrary NetBIOS name. Therefore, it is possible to derive a NetBIOS name from a given Internet host name, but not necessarily vice versa.

NetBIOS is used to resolve names within a local subnet, and is also used to resolve names within a larger network using a name server. However, it is only defined for IPv4. As such, its use for name resolution has largely been superseded by newer protocols, such as the Link-Local Multicast Name Resolution (LLMNR) Protocol [RFC4795] and the Domain Name System (DNS) [RFC1035].

1.5 Prerequisites/Preconditions

The prerequisites and preconditions are unchanged from [RFC1001] and [RFC1002].

1.6 Applicability Statement

This extension is applicable for discovering the IPv4 addresses of resources.

1.7 Versioning and Capability Negotiation

There is no versioning or localization support in this extension.

1.8 Vendor-Extensible Fields

It is important to understand that the choice of name used by a higher-layer protocol or application is up to that protocol or application and not NetBIOS. As such, this section provides a convention for use by higher-layer protocols and applications, but the extensions in this document do not enforce the use of this convention.

The recommended convention is for higher-layer protocols and applications to use the first 15 bytes of the Internet host name of the machine (padded with spaces if shorter than 15 bytes) followed by a 1-byte NetBIOS suffix chosen by the higher-layer protocol or application.<1>

The recommended convention allows for 256 NetBIOS suffix values and vendors can define a new value. However, there is no mechanism to acquire a unique value and hence collisions are possible if multiple vendors define the same NetBIOS suffix values. It is up to each higher-layer protocol or application to specify what NetBIOS suffix it uses, or how the NetBIOS name is constructed if it does not use this recommended convention.

1.9 Standards Assignments

None beyond what is in [RFC1001], [RFC1002], and [RFC2132] section 8.7.

2 Messages

2.1 Transport

The transport is unchanged from [RFC1002] except that name resolution is supported only over UDP and not TCP. The term "NetBIOS over TCP" refers to the standard protocol in the same way as [RFC 1001] and [RFC 1002] do; that is, "TCP" refers to "TCP/IP".

2.2 Message Syntax

2.2.1 NetBIOS Name Syntax

[RFC1001] and [RFC1002] are confusing with respect to the definition of the name syntax. [RFC1001] section 5.2 states: "The name space is flat and uses sixteen alphanumeric characters. Names may not start with an asterisk (*)."

[RFC1002] section 4.1 states: "The following is the uncompressed representation of the NetBIOS name "FRED", which is the 4 ASCII characters, F, R, E, D, followed by 12 space characters (0x20)."

It should be clear from the previous statement, because an asterisk and space characters are not letters or numbers, that the term "alphanumeric characters" is confusing at best.

This document clarifies the ambiguity by specifying that the name space is defined as sixteen 8-bit binary bytes, with no restrictions, except that the name SHOULD NOT<2><3> start with an asterisk (*).

Neither [RFC1001] nor [RFC1002] discusses whether names are case-sensitive. This document clarifies this ambiguity by specifying that because the name space is defined as sixteen 8-bit binary bytes, a comparison MUST be done for equality against the entire 16 bytes. As a result, NetBIOS names are inherently case-sensitive.

It is important to understand that the choice of name used by a higher-layer protocol or application is up to that protocol or application and not NetBIOS. A NetBIOS over TCP implementation MUST NOT enforce the use of the convention discussed in section 1.8.

2.2.2 MULTIHOMED NAME REGISTRATION REQUEST

[RFC1002] section 4.2.2 defines the format of a NAME REGISTRATION REQUEST. This extension adds a MULTIHOMED NAME REGISTRATION REQUEST with an identical format except that the OPCODE field MUST be set to 0xF (15).

3 Protocol Details

3.1 NetBIOS End Node Details

3.1.1 Abstract Data Model

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

The data model is as specified in [RFC1002] and [HYBRID], with the following clarifications.

- **Interface List:** A list of interfaces on which NetBIOS over TCP is enabled, in order from most preferred to least preferred. This list SHOULD be administratively configurable. Each entry contains the following:
  - **NetBIOS Name Service:** A configured list of NetBIOS name server addresses, in order from most preferred to least preferred.
- **Node Type:** The NetBIOS node type as specified in [RFC1002] and [HYBRID]. This document clarifies that this state is global, not per-interface.
- **Query Table:** For each outstanding resolution in progress, in addition to the information specified in [RFC1002] and [HYBRID], the following field is kept:
  - **Interface List:** A list of interfaces awaiting a response.
- **Local Name Table:** The local name table as specified in [RFC1002] and [HYBRID], with the following clarifications:
  - **Interface List:** A list of interfaces on which registration was attempted. Each entry also contains:
    - **Conflict Detected Flag:** A flag that, if TRUE, indicates that a conflict was seen. This document clarifies that this flag is kept per-interface (not globally) for each local name.
  - **Refresh Timeout:** [RFC1002] section 5.1.2.1 specifies the use of a Refresh Timeout. This document clarifies that this state is global, not per-interface.
- **ReadLMHostsFile:** A Boolean value defaulted to FALSE.<4> If TRUE, the **LMHOSTS** file will be read if the LMHOSTS file exists.

3.1.2 Timers

There is an lmhost_include timer used to determine whether an LMHOSTS file can be read (see section 3.1.6).

There are timers specified in [RFC1002] and [HYBRID], except as follows.

[RFC1002] section 6 states that UCAST_REQ_RETRY_TIMEOUT "should" be 5 seconds but that an adaptive timer "may" be used. In these extensions, UCAST_REQ_RETRY_TIMEOUT SHOULD be set to 1.5 seconds.

[RFC1002] specifies the use of a periodic REFRESH_TIMER for each entry in the Local Name Table, with a period of Refresh Timeout. This extension clarifies that the REFRESH_TIMER for each name is kept globally, not per-interface.

3.1.3 Initialization

The rules for initialization are specified in [RFC1002] and [HYBRID]. However, they are ambiguous as to how an end node chooses a node type. This document clarifies the rules as follows.

The NetBIOS Node Type (see [RFC1001] section 10 and [HYBRID]) SHOULD be administratively configurable, and be set to H by default.

If DHCP is in use and a NetBIOS over TCP/IP Node Type Option (see [RFC2132] section 8.7) is provided by the DHCP server, an end node MUST set its Node Type to the value indicated in the option. If this DHCP option is obtained over multiple interfaces, then the end node MUST choose one of them in any implementation-specific<5> way.

If **ReadLMHostsFile** is TRUE and if the LMHOSTS file exists, the LMHOSTS file will be read at NetBIOS initialization, and any entries marked with #PRE will be loaded into the local name table (see section 3.1.8). <6>

3.1.4 Higher-Layer Triggered Events

Except as specified in the following sections, the rules for handling higher-layer triggered events are as specified in [RFC1002] section 5.1, and [HYBRID]. This document clarifies that whenever [RFC1002] or [HYBRID] specify that a packet is to be broadcast, the end node MUST broadcast the packet on all interfaces in its Interface List unless otherwise specified.

3.1.4.1 Registering a NetBIOS Name

When a higher-layer protocol or application requests that a NetBIOS name be registered on a given interface, processing MUST be done as specified in [RFC1002] section 5.1 and [HYBRID] according to its Node Type, except as follows.

If the name already exists in the Local Name Table and the Conflict Detected flag is set on any interface in the Interface List, then the node MUST immediately fail the request.

If the end node is **multihomed**, the name to be registered is unique, and the end node is configured with a NetBIOS name server, then the end node SHOULD send a MULTIHOMED NAME REGISTRATION REQUEST (UNIQUE).

If the registration completes successfully and no entry exists in the Local Name Table, then one MUST be added with the Interface List set to contain the given interface, with its Conflict Detected flag cleared. The Refresh Timeout MUST be set to the TTL in the POSITIVE NAME REGISTRATION RESPONSE, or to 5 minutes if the TTL is less than 5 minutes.

If the registration completes successfully and an entry already exists in the Local Name Table, then the given interface MUST be added to the entry's Interface List, with its Conflict Detected flag cleared. The Refresh Timeout MUST then be set, unless its value would increase by doing so, to the TTL in the POSITIVE NAME REGISTRATION RESPONSE, or to 5 minutes if the TTL is less than 5 minutes.

If the registration fails and an entry already exists in the Local Name Table, then the given interface MUST be added to the entry's Interface List, with its Conflict Detected flag set.

If the name begins with an asterisk (*), then the request MUST be completed successfully without attempting to register the name or check for conflicts.

3.1.4.2 Resolving a NetBIOS Name

The rules for resolving a NetBIOS name are unchanged from [RFC1002] section 5.1 and [HYBRID] except as follows.

For Node Types other than B, the list of NetBIOS name servers to use MUST first be constructed as specified in section 3.1.4.2.1.

Name queries MUST then be performed as specified in [RFC1002] section 5.1 and [HYBRID], except that instead of simply querying a single NetBIOS name server, each name server in the list of NetBIOS name servers **(NBNS)** MUST be consulted in turn until one responds or the end of the list is reached.

If the end of the list of NBNS is reached, the Name Query MAY be processed against a local file LMHOSTS (see section 3.1.8).<7>

3.1.4.2.1 NetBIOS Name Server Selection

If the application or higher-layer protocol specified a specific interface, then the NetBIOS name server list MUST be the list of name servers for that interface.

If the application or higher-layer protocol did not specify a specific interface, then the NetBIOS name server list MUST be formed by concatenating the lists of name servers for each interface in the Interface List, in the order the interfaces appear in the Interface List.

3.1.5 Message Processing Events and Sequencing Rules

The rules for processing NetBIOS messages are unchanged from [RFC1002] section 5.1 and [HYBRID]. This document clarifies that whenever [RFC1002] or [HYBRID] specify that a packet is to be broadcast, the end node MUST broadcast the packet on all interfaces in its Interface List, unless otherwise specified. In addition, whenever [RFC1002] or [HYBRID] state that the Conflict Detected flag is set, this refers to the Conflict Detected flag for the interface over which the relevant message was received, unless otherwise specified.

3.1.5.1 Handling a NAME REGISTRATION REQUEST

[RFC1002] section 5.1 and [HYBRID] are somewhat confusing as to how a node is to respond to a NAME REGISTRATION REQUEST when the name matches an entry in the local name table with the Conflict Detected flag set. For example, [RFC1002] section 5.1.1.5 states that a NEGATIVE NAME REGISTRATION RESPONSE is sent if an entry exists in the local name table. However, it later clarifies that a name in the state "conflict detected" does not "logically" exist on that node and that such an entry will not be used for purposes of processing incoming request packets.

This document clarifies that a node MUST NOT send a NEGATIVE NAME REGISTRATION RESPONSE if there exists any entry in the name's Interface List whose Conflict Detected flag is set, independent of the interface on which the NAME REGISTRATION REQUEST was received.

In addition, the node MUST NOT send a NEGATIVE NAME REGISTRATION RESPONSE if the name begins with an asterisk ("*").

3.1.6 Timer Events

In addition to the timer events specified in [RFC1002] and [HYBRID], there is an lmhost_include timer event related to reading LMHOSTS files. Before attempting to open an LMHOSTS file during "#BEGIN ALTERNATE" file processing, the lmhost_include timer is initialized to NNN seconds. If the lmhost_include timer expires and raises an event, the file open is abandoned (see Alternate Block Processing in section 3.1.8).

3.1.7 Other Local Events

When an address change occurs on an interface, then the node MUST do the following.

For each entry in the Local Name Table, if the interface is in the entry's Interface List, then the node MUST repeat the registration of that name on that interface, and update the interface's Conflict Detected flag to be clear if it completes successfully, or set if it fails.

3.1.8 Using the LMHOSTS File to Resolve a Name Query

The LMHOSTS file is read on two separate occasions. At the initialization of the NetBIOS system, LMHOSTS is read to initialize the local name cache with the entries that are labeled "#PRE". During NetBIOS name resolution, if the name cannot be resolved from the local name cache or by using the normal NetBIOS protocol name resolutions, then the NetBIOS name resolution process system can be configured to scan the LMHOSTS file on a per-query basis, looking for entries that resolve the query. <8>

The LMHOSTS file is a static text file of NetBIOS name and IPv4 addresses along with additional directives for processing, including a "#include <filename>" mechanism.

- There can be one entry per line. An entry consists of an IPv4 address and a name, which can be either a computer name or a NetBIOS service name.
- Comment lines in the LMHOSTS file start with "#".
- Comments can start after the start of a line, with "#", and without the use of LMHOSTS keywords. (See the LMHOSTS keywords in the table that follows.)
- ComputerName Entries consist of an IPv4 address and a NetBIOS computer name where the name is 1 to 15 characters in length. A computer name can be used to either: 1) resolve a name to an IP address, or 2) resolve a NetBIOS service name to an address. Example: "131.107.7.29 emailsrv1".
- ServiceName entries consist of an IPv4 address and a NetBIOS service name that specifies a 16-byte name where the last byte indicates the type of the service and bytes 1 to 15 specify ComputerName, padded at the end with blanks to the 15th byte:
  - 131.107.7.30 "ComputerName 0x03" where the last byte is specified in hex.
- Entry Names are not case-sensitive.
- The LMHOSTS file has an implementation-specific file location.<9>

When NetBIOS name resolution, which uses NetBIOS protocols, does not result in successful name resolution, the resolution process can attempt to use a file-based resolution. If **ReadLMHostsFile** is TRUE and if the LMHOSTS file exists, the ComputersQuery will read the LMHOSTS file for NetBIOS name resolution to an IPv4 address.

When resolving a name, the LMHOSTS file is opened and read, and a search is made for a matching entry, or entries, so that name resolution can return a list of IP addresses. The LMHOSTS file MUST be read sequentially, matching the name in the query with the name of an entry read from the LMHOSTS file, until all matches are found or no additional entries are in the LMHOSTS file. The matching function works as follows:

1. Create an empty list of matching IP addresses, and loop for each entry.
2. If the 16th and last byte of a queryname is 0x00, 0x03 or 0x20, then queryname is stripped of its 16th byte and any trailing spaces; this results in ComputerName. If the resulting ComputerName and the name in the entry match, then add the entry's IP address to the list of matching IP addresses. If there is no #MH tag on the entry, then DONE.
3. If queryname is 16 bytes, entry name is 16 bytes, and there is an exact match, then there is a successful match. If the resulting ComputerName and the name in the entry match, then add the entry's IP address to the list of matching IP addresses. If there is no #MH tag on the entry, then DONE.
4. If not DONE, and there are more entries to be read, loop to the preceding item two.

Name resolution returns the list of matching IP addresses.

The #include facility in LMHOSTS can result in reading additional files.<10>

Predefined Keywords in LMHOSTS file

The LMHOSTS file can contain predefined keywords that are prefixed with the "#" character. The following LMHOSTS keywords table lists possible LMHOSTS keywords.

LMHOSTS Keyword	Description
#PRE	A tag that can follow the name in an entry. Tagged entries SHOULD be loaded as permanent entries in the NetBIOS name cache during initialization of the NetBIOS name system. Preloaded entries are used to reduce network broadcasts. An entry tagged with #PRE will be loaded in the **NetBT** Local Name table.
#DOM:DomainName	A tag that can follow the name in an entry. It identifies Domain Controllers for the domain *DomainName*.
#NOFNR	A tag that can follow the name of an entry. It directs NetBIOS to keep this name in the NetBT local name cache. The local name entry is marked in the Unicast field to be FALSE, to suppress any use of NetBIOS unicast name queries for older computers running LAN Manager for UNIX. <11>
#INCLUDE Path\FileName	Reads entries in the file "Path\FileName". FileName conforms to the **Universal Naming Convention (UNC)** path such as "\\fileserver1\public". There must be entries for the computer names of remote servers hosting the shares in the local LMHOSTS file; otherwise, the shares will not be accessible.
#BEGIN_ALTERNATE and #END_ALTERNATE	A tag defines a list of alternative locations for the LMHOSTS files. This is used as a reliability mechanism. Only one of the files in a "#BEGIN"/"#END" block will be used. An attempt is made to read a file, one file at a time.
#MH	A tag that can follow the name of an entry. If present, that name can have multiple IP addresses reflected in multiple entries with the same IP address. This allows the reading of an LMHOSTS file to continue after a successful match of an entry.

Because the LMHOSTS file is read sequentially, the most-frequently accessed computers SHOULD be the first entries of the file, and the #PRE-tagged entries as the last entries of the file.<12>

Using a #include LMHOSTS File

NetBT can read LMHOSTS files that are located on other computers. This allows the use of a centralized LMHOSTS file that can be accessed through a computer's local LMHOSTS file. Using a centralized LMHOSTS file still requires each computer to have a local LMHOSTS file.

To access a centralized LMHOSTS file, a computer's local LMHOSTS file MUST have an entry with the #INCLUDE tag for the location of the centralized file. Example:

#INCLUDE \\Fileserver\Public\Lmhosts

In this example, NetBT includes the LMHOSTS file on the public shared folder of the server.

The NetBT system MAY read the centralized LMHOSTS file before a user logs on to the computer as part of NetBT initialization.<13><14><15>

Alternate Block Processing

The #BEGIN_ALTERNATE and #END_ALTERNATE tags allow a block of remote LMHOSTS file locations in the reading of the LMHOSTS file. This technique is known as block inclusion. Example:

#BEGIN_ALTERNATE

#INCLUDE \\Bootsrv3Fileserver\Public\Lmhosts

#INCLUDE \\Bootsrv4Fileserver\Public\Lmhosts

#INCLUDE \\Bootsrv9Feilserver\Public\Lmhosts

#END_ALTERNATE

The files inside an ALTERNATE block are opened and read one at a time and in order. When opening a file, the include_file timer is initialized to NNN minutes. If the open succeeds, the timer is canceled and the file is read and processed. If the name resolution process is not successfully completed while reading the file, the file is closed and the #include is processed. If an include_file timer expires, the open will be canceled and the processing will move onto the next #INCLUDE in the ALTERNATE block. If there are no #INCLUDE lines left, the processing will start on the next line in the original LMHOSTS file. By this process, if any of the files are available to be read, the subsequent files in the ALTERNATE block are not read.

Creating Lmhosts Entries for Specific NetBIOS names

This form of specifying a ComputerName entry allows for resolution of the common NetBIOS service names:

- <ComputerName>0x00
- <ComputerName>0x03
- <ComputerName>0x20

In this example <ComputerName> is 15 bytes in length. A short computer name SHOULD be padded out to 15 bytes with spaces. These names correspond to the Workstation (0x00), Server (0x03), and Messenger services (0x20), respectively.

However, you might need to resolve a specific 16-character NetBIOS name to a NetBIOS application running on a remote computer. You can configure any arbitrary 16-byte NetBIOS name in the LMHOSTS file by using the following syntax:

IPv4Address "<Name><SpacePadding>\0x NN "

In which:

- IPv4Address is the IPv4 address to which this NetBIOS name is resolved
- <Name>is the first part of the NetBIOS name (up to 15 bytes)
- <SpacePadding> is needed to ensure that the full NetBIOS name is 16 bytes. If the Name portion has fewer than 15 bytes, it MUST be padded with spaces up to 15 bytes.
- 0xNN indicates the two-digit hexadecimal representation of the 16th byte of the NetBIOS name. The syntax \0xNN can represent any byte in the NetBIOS name but is most often used for the 16th character.

3.2 NetBIOS Name Server Details

3.2.1 Abstract Data Model

The data model is unchanged from [RFC1002] and [HYBRID]. This document clarifies that a name server MUST support storing at least 25 addresses per NetBIOS name.

3.2.2 Timers

None beyond what is specified in [RFC1002] and [HYBRID].

3.2.3 Initialization

The rules for initialization are unchanged from [RFC1002] and [HYBRID].

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

Except as specified in the following sections, the rules for processing NetBIOS messages are unchanged from [RFC1002] section 5.1.4.

3.2.5.1 Handling a NAME REGISTRATION REQUEST (GROUP)

When a name server receives a NAME REGISTRATION REQUEST for a group name, the server MUST handle it as specified in [RFC1002] section 5.1.4.1, except as follows.

If an entry exists for a group name, then the name server SHOULD<16> skip the step of removing any previously stored address, so that the new address gets appended to the list rather than replacing it; the name server may choose to store the broadcast address 255.255.255.255 instead.

When appending a new address to an existing list, if the list would become larger than the maximum number of entries per name supported by the name server, the name server MUST remove the oldest stored address and then append the new address. (This is consistent with the

[RFC1002] behavior, except that an implementation of [RFC1002] without these extensions only supports a maximum of a single address.)

3.2.5.2 Handling a MULTIHOMED NAME REGISTRATION REQUEST (GROUP)

When a name server receives a MULTIHOMED NAME REGISTRATION REQUEST for a group name, the server MUST handle it the same as a NAME REGISTRATION REQUEST for a group name.

3.2.5.3 Handling a MULTIHOMED NAME REGISTRATION REQUEST (UNIQUE)

When a name server receives a MULTIHOMED NAME REGISTRATION REQUEST for a **unique name**, the server MUST handle it the same as a NAME REGISTRATION REQUEST for a unique name as specified in [RFC1002] section 5.1.4.1, except as follows.

Instead of removing an address when one is already stored, the oldest address MUST only be removed when the maximum number of addresses per name would be exceeded.

3.2.6 Timer Events

None beyond what is specified in [RFC1002] and [HYBRID].

3.2.7 Other Local Events

None beyond what is specified in [RFC1002] and [HYBRID].

4 Protocol Examples

Consider a multihomed node A with an Internet host name of "EXAMPLE" supporting the extensions defined herein.

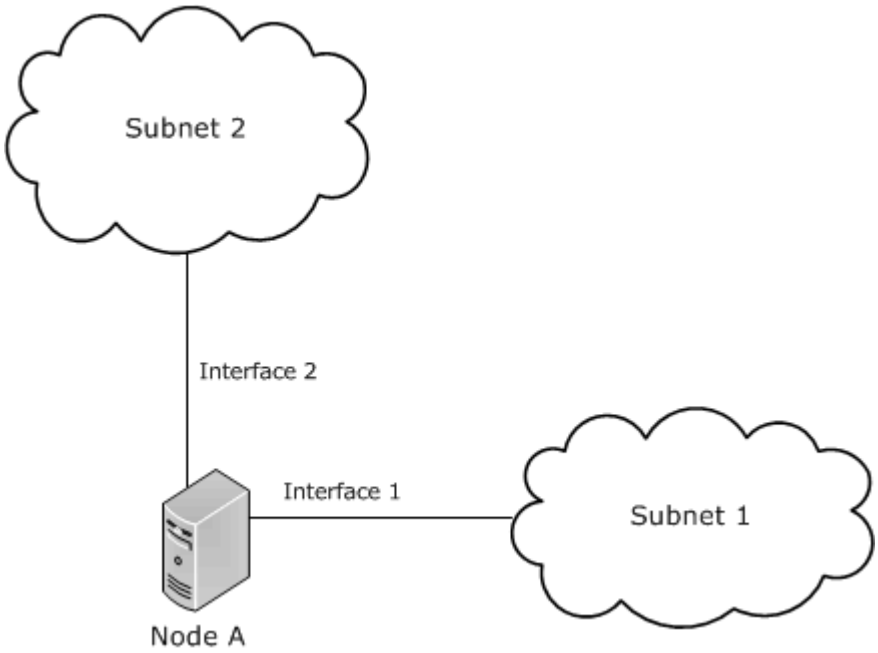

Figure 1: Multihomed node example

1. At startup, Node A uses DHCP on each interface. On interface 1, it gets a NetBIOS over TCP/IP Node Type Option indicating it should be an H node. From interface 2, it gets no DHCP response, but defaults to H node behavior. Hence when it decides to be an H node based on interface 2, it overwrites its Node Type to be the type from interface 2. However, in this example, the value is effectively unchanged because it was already an H node based on DHCP from interface 1.

2. Later, an application wants to publish a NetBIOS name and chooses to use the convention defined in section 1.8. The application chooses a NetBIOS suffix of 0x19, and constructs the NetBIOS name of "EXAMPLE", padded with spaces to 15 bytes, and puts the NetBIOS suffix in the 16th byte, resulting in the following hexadecimal bytes: [0x45, 0x58, 0x41, 0x4D, 0x50, 0x4c, 0x45, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x19]. The application asks NetBIOS to register the name EXAMPLE on interface 1, and NetBIOS sends a NAME REGISTRATION REQUEST to the first name server learned on that interface.

   The name server responds with a NEGATIVE NAME REGISTRATION RESPONSE.

   The node receives the response, and EXAMPLE is not added to the Local Name Table, and a failure is returned to the application.

3. The application then tries to register the NetBIOS name EXAMPLE on interface 2.

   No name servers are known on interface 2, so the node falls back to broadcast, and registration succeeds.

   The name EXAMPLE is then added to the Local Name Table, with interface 2 in the Interface List.

4. Later, another application tries to register the NetBIOS name EXAMPLE on interface 2. The name is already registered on that interface, so a success is immediately returned to the application.

5. Later, another application tries to register EXAMPLE on interface 1.

   The name server again responds with a NEGATIVE NAME REGISTRATION RESPONSE causing registration to again fail on subnet 1.

   This time, because an entry already exists in the Local Name Table, interface 1 is added to the Interface List for the name EXAMPLE in the Local Name Table, with the Conflict Detected flag set.

6. Later, another application tries to register EXAMPLE on interface 1.

   Registration fails immediately (without sending any request) because an entry already exists with the Conflict Detected flag set for some interface (interface 1).

7. Later, another application tries to register EXAMPLE on interface 2.

   Registration fails immediately (without sending any request) because an entry already exists with the Conflict Detected flag set for some interface (interface 1).

8. Later, a NAME QUERY is received on interface 2 for the name EXAMPLE.

   The node replies with a POSITIVE NAME QUERY RESPONSE because the Conflict Detected flag is clear for that interface.

9. Later, a NAME QUERY is received on interface 1 for the name EXAMPLE.

   No response is sent because the Conflict Detected flag is set for that interface.

10. Later, a NAME REGISTRATION REQUEST is received for the name EXAMPLE on interface 1.

    No response is sent because the Conflict Detected flag is set on some interface (interface 1).

11. Later, a NAME REGISTRATION REQUEST is received for the name EXAMPLE on interface 2.

    No response is sent because the Conflict Detected flag is set on some interface (interface 1).

12. Later, an address change occurs on interface 1 after the conflicting entry has been removed from the name server by an administrator.

    Node A tries to reregister the name EXAMPLE on interface 1 by sending a new NAME REGISTRATION REQUEST.

    This time registration succeeds and the server replies with a POSITIVE NAME REGISTRATION RESPONSE.

    The node receives the response and clears the Conflict Detected flag for interface 1.

13. Later, a NAME REGISTRATION REQUEST is received for the name EXAMPLE on interface 1.

    The node replies with a NEGATIVE NAME REGISTRATION RESPONSE because the name is in the Local Name Table and no Conflict Detected flag is set.

14. Later, a NAME REGISTRATION REQUEST is received for the name EXAMPLE on interface 2.

    The node replies with a NEGATIVE NAME REGISTRATION RESPONSE because the name is in the Local Name Table and no Conflict Detected flag is set.

5 Security Considerations

5.1 Security Considerations for Implementers

The security considerations are unchanged from [RFC1001] and [RFC1002].

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

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

- Microsoft Windows NT® operating system
- Microsoft Windows® 2000 operating system
- Windows® XP operating system
- Windows Server® 2003 operating system
- Windows Vista® operating system
- Windows Server® 2008 operating system
- Windows® 7 operating system
- Windows Server® 2008 R2 operating system

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

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

<1> Section 1.8: Unless a protocol using the recommended convention specifies otherwise, Windows protocols use the machine system **code page** for NetBIOS names; this means that two computers with different code pages cannot interoperate by using such a protocol with anything other than ASCII names.

<2> Section 2.2.1: All Windows versions do not enforce the asterisk restriction when a name is queried.

<3> Section 2.2.1: Windows 2000, Windows XP, and Windows Server 2003 allow the SMB protocol to register the name "*SMBSERVER".

<4> Section 3.1.1: **ReadLMHostsFile** has a default value of TRUE in Windows 2000, Windows XP, and Windows Server 2003.

<5> Section 3.1.3: When a DHCP option is received, Windows stores the value in its Node Type, so that the most recent one received is the one used for subsequent operations.

<6> Section 3.1.3: Windows 7, Windows Server 2008 R2, Windows Vista, and Windows Server 2008 do not read the LMHOSTS file by default.

<7> Section 3.1.4.2: Microsoft does not recommend using LMHOSTS. Windows 7, Windows Server 2008 R2, Windows Vista, and Windows Server 2008 do not perform LMHOSTS-based name resolution by default.

<8> Section 3.1.8: Microsoft has deprecated the use of the LMHOSTS file and does not recommend its use for name resolution. Windows 7, Windows Server 2008 R2, Windows Vista, and Windows

Server 2008 do not use LMHOSTS by default. Windows 2000, Windows XP, and Windows Server 2003 will use LMHOSTS file resolution if the LMHOSTS file is created and if all other forms of name resolution fail, including DNS and the NetBIOS name resolution protocols.

<9> Section 3.1.8: On Windows, this file is in the systemroot\System32\Drivers\Etc folder.

<10> Section 3.1.8: By default, LMHOSTS is not used in Windows 7, Windows Server 2008 R2, Windows Vista, or Windows Server 2008.

<11> Section 3.1.8: This is a legacy behavior in pre-Windows 2000 servers.

<12> Section 3.1.8: Because the #PRE entries are loaded by Windows into the NetBIOS name cache, they are not needed when NetBT reads the LMHOSTS file after startup. Placing them as the last entries of the file allows NetBT to scan the LMHOSTS file for other NetBIOS names; a successful match will eliminate the need to continue reading the file and will eliminate #PRE entry processing of those entries at the end of the file.

<13> Section 3.1.8: On Windows, because no user name is associated with the computer during startup, NetBT uses a null user name for its credentials when accessing the shared folder where the central LMHOSTS file is located.

<14> Section 3.1.8: Windows 7, Windows Server 2008 R2, Windows Vista, and Windows Server 2008 do not read the LMHOSTS file by default.

<15> Section 3.1.8: To allow null access to a shared folder that contains an LMHOSTS file on a Windows machine, the name of the folder should be set for the registry value of HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet \Services\Lanmanserver \Parameters\NullSessionShares.

<16> Section 3.2.5.1: Windows follows the RFC behavior of replacing addresses if the last byte of the group name is not 0x1c. For group names with the last byte equal to 0x1c, Windows appends addresses. For any group names except those with a last byte of 0x20 or 0x1c, Windows returns 255.255.255.255 in response to queries as if it had stored 255.255.255.255.

7 Change Tracking

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

8 Index

A

Abstract data model
  end node 9
  name server 15
Applicability 6

C

Capability negotiation 7
Change tracking 22

D

Data model - abstract
  end node 9
  name server 15

E

End node
  abstract data model 9
  higher-layer triggered events
    overview 10
    registering NetBios name 10
    resolving NetBios name 11
  initialization 10
  local events 12
  message processing
    handling NAME REGISTRATION REQUEST 11
    overview 11
  sequencing rules
    handling NAME REGISTRATION REQUEST 11
    overview 11
  timer events 12
  timers 9
Examples - overview 17

F

Fields - vendor-extensible 7

G

Glossary 5

H

Higher-layer triggered events
  end node
    overview 10
    registering NetBios name 10
    resolving NetBios name 11
  name server 15

I

Implementer - security considerations 19
Index of security parameters 19
Informative references 6
Initialization
  end node 10
  name server 15
Introduction 5

L

Local events
  end node 12
  name server 16

M

Message processing
  end node
    handling NAME REGISTRATION REQUEST 11
    overview 11
  name server
    handling MULTIHOMED NAME REGISTRATION REQUEST (GROUP) 16
    handling MULTIHOMED NAME REGISTRATION REQUEST (UNIQUE) 16
    handling NAME REGISTRATION REQUEST (GROUP) 15
    overview 15
Messages
  syntax
    MULTIHOMED NAME REGISTRATION REQUEST 8
    NetBIOS name syntax 8
  transport 8

N

Name server
  abstract data model 15
  higher-layer triggered events 15
  initialization 15
  local events 16
  message processing
    handling MULTIHOMED NAME REGISTRATION REQUEST (GROUP) 16
    handling MULTIHOMED NAME REGISTRATION REQUEST (UNIQUE) 16
    handling NAME REGISTRATION REQUEST (GROUP) 15
    overview 15
  sequencing rules
    handling MULTIHOMED NAME REGISTRATION REQUEST (GROUP) 16
    handling MULTIHOMED NAME REGISTRATION REQUEST (UNIQUE) 16
    handling NAME REGISTRATION REQUEST (GROUP) 15
    overview 15
  timer events 16
  timers 15
NetBIOS name syntax 8

Normative references 5

O

Overview (synopsis) 6

P

Parameters - security index 19
Preconditions 6
Prerequisites 6
Product behavior 20

R

References
  informative 6
  normative 5
Registering NetBios name 10
Relationship to other protocols 6
Resolving NetBios name
  NetBIOS name server selection 11
  overview 11

S

Security
  implementer considerations 19
  parameter index 19
Sequencing rules
  end node
    handling NAME REGISTRATION REQUEST 11
    overview 11
  name server
    handling MULTIHOMED NAME REGISTRATION REQUEST (GROUP) 16
    handling MULTIHOMED NAME REGISTRATION REQUEST (UNIQUE) 16
    handling NAME REGISTRATION REQUEST (GROUP) 15
    overview 15
Server
  abstract data model 15
  higher-layer triggered events 15
  initialization 15
  local events 16
  message processing
    handling MULTIHOMED NAME REGISTRATION REQUEST (GROUP) 16
    handling MULTIHOMED NAME REGISTRATION REQUEST (UNIQUE) 16
    handling NAME REGISTRATION REQUEST (GROUP) 15
    overview 15
  sequencing rules
    handling MULTIHOMED NAME REGISTRATION REQUEST (GROUP) 16
    handling MULTIHOMED NAME REGISTRATION REQUEST (UNIQUE) 16
    handling NAME REGISTRATION REQUEST (GROUP) 15
    overview 15
  timer events 16
  timers 15
Standards assignments 7
Syntax
  MULTIHOMED NAME REGISTRATION REQUEST 8
  NetBIOS name syntax 8

T

Timer events
  end node 12
  name server 16
Timers
  end node 9
  name server 15
Tracking changes 22
Transport 8
Triggered events - higher-layer
  end node
    overview 10
    registering NetBios name 10
    resolving NetBios name 11
  name server 15

V

Vendor-extensible fields 7
Versioning 7