Network Working Group
INTERNET-DRAFT
draft-leach-cifs-browser-spec-00.txt
Category: Informational
Expires June 10, 1997

Paul J. Leach, Microsoft
Dilip C. Naik, Microsoft

January 10, 1997

CIFS/E Browser Protocol

Preliminary Draft

STATUS OF THIS MEMO

THIS IS A PRELIMINARY DRAFT OF AN INTERNET-DRAFT. IT DOES NOT REPRESENT THE CONSENSUS OF ANY WORKING GROUP.

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Distribution of this document is unlimited. Please send comments to the authors or the CIFS mailing list at <cifs@listserv.msn.com>. Discussions of the mailing list are archived at <URL:http://microsoft.ease.lsoft.com/archives/cifs.html>.

ABSTRACT

The CIFS/E (CIFS extensions for enterprise networks) family of protocols includes a protocol for *browsing*. Browsing is a mechanism for discovering servers that are running particular services (not just CIFS file services). Servers are organized into named groups called *domains,* which form browsing scopes. This document specifies version 1.15 of the browsing protocol. It also specifies the mailslot protocol, because the browsing protocol depends on it (and is the only CIFS/E protocol which does).

Table of Contents

1Introduction

The CIFS/E (CIFS extensions for enterprise networks) family of protocols includes a protocol for "*browsing*". Browsing is a mechanism for discovering servers that are running particular services (not just CIFS file services). Servers are organized into named groups called "*domains*", which form browsing scopes. This document specifies version 1.15 of the browsing protocol. It also specifies the mailslot protocol, because the browsing protocol depends on it (and is the only CIFS/E protocol which does).

This document uses the traditional RFC keywords MUST, SHOULD, etc., (now documented in [Bradner 96]) to indicate requirement levels for interoperability.

Note: This document is about CIFS/E browsers and has nothing to do whatsoever with Web browsers such as Internet Explorer and Netscape Navigator. This is a specification for persons interested in implementing a browser server or client that can inter-operate with other CIFS/E browsers and clients.

2Prerequisites

- Familiarity with Common Internet File System specification (CIFS) in general and the Transact2 SMB as well as Remote Administration Protocol in particular [CIFS 96]
- Familiarity with concepts of subnets and NETBIOS [RFC 1001].

Additional information about browsing may be found in the MSDN articles “Browsing and Windows 95” Parts I, II and III. These articles cover considerations for browser deployment, especially in a WAN environment.

3Browser Overview

Hosts involved in the browsing process can be separated into two distinct groups, browser clients and browser servers (often referred to simply as “*browsers*”).

A browser is a server which maintains information about servers – primarily the domain they are in and the services that they are running -- and about domains. Browsers may assume several different roles in their lifetimes, and dynamically switch between them.

Browser clients are of two types: workstations and (non-browser) servers. In the context of browsing, workstations query browsers for the information they contain; servers supply browsers the information by registering with them. Note that, at times, browsers may themselves behave as browser clients and query other browsers.

For the purposes of this specification, a domain is simply a name with which to associate a group of resources such as computers, servers and users. Domains allow a convenient means for browser clients to restrict the scope of a search when they query browser servers. Every domain has a “master” server called the Primary Domain Controller (PDC) that manages various activities within the domain.

One browser for each domain on a subnet is designated the Local Master Browser for that domain. Servers in its domain on the subnet register with it, as do the Local Master Browsers for other domains on the subnet. It uses these registrations to maintain authoritative information about its domain on its subnet. If there are other subnets in the network, it also knows the name of the server running the domain's Domain Master Browser; it registers with it, and uses it to obtain information about the rest of the network (see below).

Clients on a subnet query browsers designated as the Backup Browsers for the subnet (not the Master Browser). Backup Browsers maintain a copy of the information on the Local Master Browser; they get it by periodically querying the Local Master Browser for all of its information. Clients find the Backup Browsers by asking the Local Master Browser. Clients are expected to spread their queries evenly across Backup Browsers to balance the load.

The Local Master Browser is dynamically elected automatically. Multiple Backup Browser Servers may exist per subnet; they are selected from among the potential browser servers by the Local Master Browser, which is configured to select enough to handle the expected query load.

When the re are multiple subnets, a Domain Master Browser is assigned the task of keeping the multiple subnets in synchronization. The Primary Domain Controller (PDC) always acts as the Domain Master Browser. The Domain Master Browser periodically acts as a client and queries all the Local Master Browsers for its domain, asking them for a list containing all the domains and all the servers in their domain known within their subnets; it merges all the replies into a single master list. This allows a Domain Master Browser server to act as a collection point for inter-subnet browsing information. Local Master Browsers periodically query the Domain Master Browser to retrieve the network-wide information it maintains.

When a domain spans only a single subnet, there will not be any distinct Local Master Browser; this role will be handled by the Domain Master Browser. Similarly, the Domain Master Browser is always the Local Master Browser for the subnet it is on.

When a browser client suspects that the Local Master Browser has failed, the client will instigate an election in which the browser servers participate, and some browser servers may change roles.

Some characteristics of a good browsing mechanism include:
- minimal network traffic
- minimum server discovery time
- minimum change discovery latency
- immunity to machine failures

Historically, Browser implementations had been very closely tied to NETBIOS and datagrams. The early implementations caused a lot of broadcast traffic. See Appendix D for an overview that presents how the Browser specification evolved.

4Browsing Protocol Architecture

This section first describes the how the browsing protocol is layered, then describes the roles of clients, servers, and browsers in the browsing subsystem.

4.1Layering of Browsing Protocol Requests

Most of the browser functionality is implemented using mailslots. Mailslots provide a mechanism for fast, unreliable unidirectional data transfer; they are named via ASCII “mailslot (path) name”. Mailslots are implemented using the CIFS Transact SMB which is encapsulated in a NETBIOS datagram. Browser protocol requests are sent to browser specific mailslots using some browser-specific NETBIOS names. These datagrams can either be unicast or broadcast, depending on whether the NETBIOS name is a “unique name” or a “group name”. Various data structures, which are detailed subsequently within this document, flow as the data portion of the Transact SMB.

Here is an example of a generic browser SMB, showing how a browser request is encapsulated in a TRANSACT SMB request. Note that the PID, TID, MID, UID, and Flags are all 0 in mailslot requests.

```
SMB: C transact, File = \MAILSLOT\BROWSE
  SMB: SMB Status = Error Success
    SMB: Error class = No Error
    SMB: Error code = No Error
  SMB: Header: PID = 0x0000 TID = 0x0000 MID = 0x0000 UID = 0x0000
    SMB: Tree ID   (TID) = 0 (0x0)
    SMB: Process ID  (PID) = 0 (0x0)
    SMB: User ID   (UID) = 0 (0x0)
    SMB: Multiplex ID (MID) = 0 (0x0)
    SMB: Flags Summary = 0 (0x0)
  SMB: Command = C transact
    SMB: Word count = 17
    SMB: Word parameters
    SMB: Total parm bytes = 0
    SMB: Total data bytes = 33
    SMB: Max parm bytes = 0
    SMB: Max data bytes = 0
    SMB: Max setup words = 0
    SMB: Transact Flags Summary = 0 (0x0)
      SMB: ..............0 = Leave session intact
      SMB: ..............0. = Response required
    SMB: Transact timeout = 0 (0x0)
    SMB: Parameter bytes = 0 (0x0)
    SMB: Parameter offset = 0 (0x0)
    SMB: Data bytes = 33 (0x21)
    SMB: Data offset = 86 (0x56)
    SMB: Setup word count = 3
    SMB: Setup words
    SMB: Mailslot opcode = Write mailslot
    SMB: Transaction priority = 1
    SMB: Mailslot class = Unreliable (broadcast)
    SMB: Byte count = 50
    SMB: Byte parameters
    SMB: Path name = \MAILSLOT\BROWSE
    SMB: Transaction data
  SMB: Data: Number of data bytes remaining = 33 (0x0021)
```

Note the SMB command is Transact, the opcode within the Transact SMB is Mailslot Write, and the browser data structure is carried as the Transact data.
The Transaction data begins with an opcode, that signifies the operation and determines the size and structure of data that follows. This opcode is named as per one of the below:

```
HostAnnouncement            1
AnnouncementRequest         2
RequestElection             8
GetBackupListReq            9
GetBackupListResp           10
BecomeBackup                11
DomainAnnouncment           12
MasterAnnouncement          13
LocalMasterAnnouncement     15
```

Browser datagrams are often referred to as simply browser frames. The frames are in particular, referred to by the name of the opcode within the Transaction data e.g. a GetBackupListReq browser frame, a RequestElection browser frame, etc.

The structures that are sent as the data portion of the Transact SMB are described in section(s) 6.2 through 6.12 in this document. These structures are tightly packed, i.e. there are no intervening pad bytes in the structure, unless they are explicitly described as being there. All quantities are sent in native Intel format and multi-byte values are transmitted least significant byte first.

Besides mailslots and Transaction SMBs, the other important piece of the browser architecture is the NetServerEnum2 request. This request that allows an application to interrogate a Browser Server and obtain a complete list of resources (servers, domains, etc) known to that Browser server. Details of the NetServerEnum2 request are presented in section 6.4. Some examples of the NetServerEnum2 request being used are when a Local Master Browser sends a NetServerEnum2 request to the Domain Master Browser and vice versa. Another example is when a browser client sends a NetServerEnum2 request to a Backup Browser server.

4.2 Browser Client

A browser client is a system running applications which may wish to query for a list of servers for a particular domain (often its own) or a list of all the domains in the network. (For example, such an application is launched when a user clicks on the Network Neighborhood icon on a Windows machine.) A browser client may send a NetServerEnum2 request (see section 6.4) to any Backup Browser serving that domain to obtain such information.

A browser client SHOULD keep a list of a few Backup Browsers for its own domain; it MAY cache lists of Backup Browsers for other domains if it browses them frequently, or it may obtain them upon demand. The objective is to minimize the cost of locating Backup Browsers each time it wants to make a NetServerEnum2 request.

A browser client SHOULD distribute its NetServerEnum2 requests randomly among all the Backup Browsers for a domain in its list. The objective is to enable multiple Backup Browsers to effectively handle high browsing loads.

A browser client SHOULD NOT send its NetServerEnum2 requests directly to a Master Browser. Browser clients unilaterally sending NetServerEnum2 requests directly to Master Browsers will result in unavoidable congestive collapse in a large enough network.

A browser client can locate browser servers for the domain it wants to browse by sending a GetBackupListRequest frame to the Local Master Browser for that domain and waiting for a GetBackupListResponse frame. See section 6.2 and section 6.3, respectively. Once the Local Master Browser server responds with a list of Backup Browser servers, the client should choose several at random from within the response, and cache them. If there is no response after a delay, the GetBackupListRequest frame may be retransmitted. The delay MUST be at least twice the expected service time, and the delay should be doubled after each time-out.

In case the Local Master Browser for a domain fails to respond to the GetBackupListRequest, the browser client may attempt to retrieve a list of Backup Browsers by sending a GetBackupListRequest frame directly to the Domain Master Browser for that domain. It can find the Domain Master Browser using the method described in appendix B.

A browser client SHOULD force an election by sending a RequestElection frame (see section 6.7) if it does not get a response to a GetBackupListRequest for its own domain after several retransmissions, since it must be assumed that the Local Master browser has crashed. Details of the election process are in sections 6.7 and 6.8.

~~A browser client that wants to issue a NetServerEnum2 request to query a domain other than its own MAY send the request to any Backup Browser in its domain if it is unable to locate Backup Browsers for the other domain (there may be none on its subnet).~~

4.3Non-Browser Server

A non-browser server is a server that has some resource(s) or service(s) it wishes to advertise as being available using the browsing protocol. Examples of non-browser servers would be an SQL server, print server, etc.

A non-browser server MUST periodically send a HostAnnouncement browser frame, specifying the type of resources or services it is advertising. Details are in section 6.5.

A non-browser server SHOULD announce itself relatively frequently when it first starts up in order to make its presence quickly known to the browsers and thence to potential clients. The frequency of the announcements SHOULD then be gradually stretched, so as to minimize network traffic. Typically, non-browser servers announce themselves once every minute upon start up and then gradually adjust the frequency of the announcements to once every 12 minutes.

A non-browser server SHOULD send a HostAnnouncement browser frame specifying a type of 0 just prior to shutting down, to allow it to quickly be removed from the list of available servers.

A non-browser server MUST receive and process AnnouncementRequest frames from the Local Master Browser, and MUST respond with a HostAnnouncement frame, after a delay chosen randomly from the interval [0,30] seconds. AnnouncementRequests typically happen when a Local Master Browser starts up with an empty list of servers for the domain, and wants to fill it quickly. The 30 second range for responses prevents the Master Browser from becoming overloaded and losing replies, as well as preventing the network from being flooded with responses.

4.4 Browser Servers

The following sections describe the roles of the various types of browser servers.

4.4.1 Potential Browser Server

A Potential Browser server is a browser server that is capable of being a Backup Browser server or Master Browser server, but is not currently fulfilling either of those roles.

A Potential Browser MUST set type SV_TYPE_POTENTIAL_BROWSER (see section 6.4.1) in its HostAnnouncement until it is ready to shut down. In its last HostAnnouncement frame before it shuts down, it SHOULD specify a type of 0.

A Potential Browser server MUST receive and process BecomeBackup frames (see section 6.9) and become a backup browser upon their receipt.

A Potential Browser MUST participate in browser elections (see section 6.8).

4.4.2 Backup Browser

Backup Browser servers are a subset of the Potential Browsers that have been chosen by the Master Browser on their subnet to be the Backup Browsers for the subnet.

A Backup Browser MUST set type SV_TYPE_BACKUP_BROWSER (see section 6.4.1) in its HostAnnouncement until it is ready to shut down. In its last HostAnnouncement frame before it shuts down, it SHOULD specify a type of 0.

A Backup Browser MUST listen for a LocalMasterAnnouncement frame (see section 6.10) from the Local Master Browser, and use it to set the name of the Master Browser it queries for the server and domain lists.

A Backup Browsers MUST periodically make a NetServerEnum2 request of the Master Browser on its subnet for its domain to get a list of servers in that domain, as well as a list of domains. The period is a configuration option balancing currency of the information with network traffic costs – a typical value is 15 minutes.

A Backup Browser SHOULD force an election by sending a RequestElection frame (see section 6.7) if it does not get a response to its periodic NetServeEnum2 request to the Master Browser.

A Backup Browser MUST receive and process NetServerEnum2 requests from browser clients, for its own domain and others. If the request is for a list of servers in its domain, or for a list of domains, it can answer from its internal lists. If the request is

for a list of servers in a domain different than the one it serves, it sends a NetServerEnum2 request to the Domain Master Browser for that domain (which it can in find in its list of domains and their Domain Master Browsers).

A Backup Browser MUST participate in browser elections (see section 6.8).

4.4.3Master Browser

Master Browsers are responsible for:

- indicating it is a Master Browser
- receiving server announcements and building a list of such servers and keeping it reasonably up-to-date.
- returning lists of Backup Browsers to browser clients.
- ensuring an appropriate number of Backup Browsers are available.
- announcing their existence to other Master Browsers on their subnet, to the Domain Master Browser for their domain, and to all browsers in their domain on their subnet
- forwarding requests for lists of servers on other domains to the Master Browser for that domain
- keeping a list of domains in its subnet
- synchronizing with the Domain Master Browser (if any) for its domain
- participating in browser elections
- ensuring that there is only one Master Browser on its subnet

A Master Browser MUST set type SV_TYPE_MASTER_BROWSER (see section 6.4.1) in its HostAnnouncement until it is ready to shut down. In its last HostAnnouncement frame before it shuts down, it SHOULD specify a type of 0.

A Master Browser MUST receive and process HostAnnouncement frames from servers, adding the server name and other information to its servers list; it must mark them as ~~"local"~~"authoritative" entries. Periodically, it MUST check all local server entries to see if a server's HostAnnouncement has timed out (no HostAnnouncement received for three times the periodicity the server gave in the last received HostAnnouncement) and remove timed-out servers from its list.

A Master Browser MUST receive and process DomainAnnouncement frames (see section 6.12) and maintain the domain names and their associated (Local) Master Browsers in its internal domain list until they time out; it must mark these as ~~"local"~~"authoritative" entries. Periodically, it MUST check all local domain entries to see if a server's DomainAnnouncement has timed out (no DomainAnnouncement received for three times the periodicity the server gave in the last received DomainAnnouncement) and remove timed-out servers from its list.

A Master Browser MUST receive and process GetBackupListRequest frames from clients, returning GetBackupListResponse frames containing a list of the Backup Servers for its domain.

A Master Browser MUST eventually send BecomeBackup frames (see section 6.9) to one or more Potential Browser servers to increase the number of Backup Browsers if there are not enough Backup Browsers to handle the anticipated query load. Note: possible good times for checking for sufficient backup browsers are after being elected, when timing out server HostAnnouncements, and when receiving a server's HostAnnouncement for the first time.

A Master Browser MUST periodically announce itself and the domain it serves to other (Local) Master Browsers on its subnet, by sending a DomainAnnouncement frame (see section 6.12) to its subnet.

A Master Browser MUST send a MasterAnnouncement frame (see section 6.11) to the Domain Master Browser after it is first elected, and periodically thereafter. This informs the Domain Master Browser of the presence of all the Master Browsers.

A Master Browser MUST periodically announce itself to all browsers for its domain on its subnet by sending a LocalMasterAnnouncement frame (see section 6.10).

A Master Browser MUST receive and process NetServerEnum2 requests from browser clients, for its own domain and others. If the request is for a list of servers in its domain, or for a list of domains, it can answer from its internal lists. Entries in its list marked ~~"local"~~"authoritative" MUST have the SV_TYPE_LOCAL_LIST_ONLY bit set in the returned results; it must be clear for all other entries. If the request is for a list of servers in a domain different than the one it serves, it sends a

NetServerEnum2 request to the Domain Master Browser for that domain (which it can in find in its list of domains and their Domain Master Browsers).

> ***Note:*** *The list of servers that the Master Browser maintains and returns to the Backup Browsers, is limited in size to 64K of data. This will limit the number of systems that can be in a browse list in a single workgroup or domain to approximately two thousand systems.*

A Master Browser SHOULD request all servers to register with it by sending an AnnouncementRequest frame, if, on becoming the Master Browser by winning an election, its server list is empty. Otherwise, clients might get an incomplete list of servers until the servers' periodic registrations fill the server list.

If the Master Browser on a subnet is not the Primary Domain Controller (PDC), then it is a Local Master Browser.

A Local Master Browser MUST periodically synchronize with the Domain Master Browser (which is the PDC). This synchronization is performed by making a NetServerEnum2 request to the Domain Master Browser and merging the results with its list of servers and domains. An entry from the Domain Master Browser should be marked "non-local", and must not overwrite an entry with the same name marked ~~"local"~~"authoritative". The Domain Master Browser is located as specified in Appendix B.

A Master Browser MUST participate in browser elections (see section 6.8).

A Master Browser for a domain "D" MUST, after winning an election, register the NetBIOS unique name D(1d).

A Master Browser for a domain "D" MUST, after losing an election, unregister the NetBIOS unique name D(1d), and do so quickly enough that the winning browser can successfully register it.

A Master Browser MUST, if it receives a HostAnnouncement, DomainAnnouncement, or LocalMasterAnnouncement frame another system that claims to be the Master Browser for its domain, demote itself from Master Browser and force an election. This ensures that there is only ever one Master Browser in each workgroup or domain.

A Master Browser SHOULD, if it loses an election, become a Backup Browser (without being told to do so by the new Master Browser). Since it has more up-to-date information in its lists than a Potential Browser, it is more efficient to have it be a Backup Browser than to promote a Potential Browser.

4.4.3.1Preferred Master Browser

A Preferred Master Browser supports exactly the same protocol elements as a Potential Browser, except as follows.

A Preferred Master Browser MUST always force an election when it starts up.

A Preferred Master Browser MUST participate in browser elections (see section 6.8).

A Preferred Master Browser MUST set the Preferred Master bit in the RequestElection frame (see section 6.7) to bias the election in its favor.

A Preferred Master Browser SHOULD, if it loses an election, automatically become a Backup Browser, without being told to do so by the Master Browser.

4.4.4Domain Master Browser

~~Since the Domain Master Browser always runs on the PDC, it must implement all the protocols required of a PDC in addition to the browsing protocol, and that is way beyond the scope of this specification~~A Domain Master Browser for a domain MUST

act as a Local Master Browser for its subnet. Thus, it acts exactly like a Local Master Browser, except where required to act differently by this section.

A Domain Master Browser MUST set type SV_TYPE_DOMAIN_MASTER (see section 6.4.1) in its HostAnnouncement until it is ready to shut down. In its last HostAnnouncement frame before it shuts down, it SHOULD specify a type of 0.

A Domain Master Browser for a domain MUST receive and process MasterAnnouncement frames from Local Master Browsers of its domain, and keep a list of all the Local Master Browsers in its domain.

A Domain Master Browser MUST periodically synchronize with the Local Master Browsers for its domain. This synchronization is performed by making a NetServerEnum2 request to each Local Master Browser for its "authoritative" entries and merging the results into a master list for the whole domain.

A Domain Master Browser MUST eventually purge an entry from its master list if:

- it was originally received from a Local Master Browser
- it has not appeared in any list obtained from a Local Master Browser for an implementation specific amount of time
- it has not received any MasterAnnouncement or HostAnnouncement for the server or domain specified by the entry

A Domain Master Browser MUST set the PDC bit in the RequestElection frame (see section 6.7) to bias the election in its favor.

A Domain Master Browser MAY be configured with a list of domains for which it is to support cross-domain browsing. It MUST periodically discover or validate the name of the Domain Master Browser for each such domain using the mechanism in appendix B, and add this information to its list of domains and their Domain Master Browsers.

5Mailslot Protocol Specification

The only transaction allowed to a mailslot is a mailslot write. Mailslot writes requests are encapsulated in CIFS TRANSACT SMBs and sent to a specified NetBIOS name. The following table shows the interpretation of the TRANSACT SMB parameters for a mailslot transaction:

Name	Value	Description
Command	SMB_COM_TRANSACTION	
Name	<name>	STRING name of mail slot to write; must start with "\MAILSLOT\"
SetupCount	3	Always 3 for mailslot writes
Setup[0]	1	Command code == write mailslot
Setup[1]	Ignored	
Setup[2]	Ignored	
TotalDataCount	n	Size of data in bytes to write to the mailslot
Data[n]		The data to write to the mailslot

When it is specified that a mailslot message is "sent to NetBIOS name X and mailslot Y" it means that a NetBIOS datagram (as specified in section 4.4.2 of [RFC 1002]) is sent whose

- MSG_TYPE is DIRECT_UNIQUE_DATAGRAM if the NetBIOS name is a unique name
- MSG_TYPE is DIRECT_GROUP_DATAGRAM if the NetBIOS name is a group name
- SOURCE_NAME field is the NetBIOS name of the sending system
- DESTINATION_NAME is X
- USER_DATA field contains an SMB_COM_TRANSACTION SMB (as specified in the CIFS spec) with its fields as specified in the table above.

(or the equivalent, if the NetBIOS service in use is over a protocol other than as specified by RFC 1001/1002.)

In order to receive mailslot messages, a system MUST have done a NetBIOS registration (as per section 5.2 of RFC 1001) of the DESTINATION_NAME to which the message was sent.

Before sending a mailslot message, the sending system SHOULD have done a NetBIOS registration of the SOURCE_NAME in the message to be sent.

6Browser Protocol Specification

As already explained, browser datagrams are also referred to as Browser frames. What distinguishes one browser ~~datagram~~frame from another is the opcode that is carried as the data portion of the Transact SMB, ~~the mailslot~~ and the NETBIOS name and mailslot to which the browser ~~datagram is sent to.~~frame is sent. Browser frames are often referred to by the symbolic name of the opcode that is within the data portion of the Transact SMB. The following sections describe the various Browser frames.

6.1NETBIOS Name Notation

NAME(xx) denotes the ASCII string "NAME," padded with spaces (0x20) to 15 bytes, with a hex xx value in the 16th byte. For example, the notation FOOBAR(15) indicates a NETBIOS name consisting of the bytes:

```
[69,79,79,65,64,82,20,20,20,20,20,20,20,20,20, 15]
```

Names that are placeholders and that need to be substituted with their actual values are bracketed within <>. Thus the string <Domain> would become “Redmond” if the domain under consideration is named “Redmond”. Details of the various NETBIOS names used for browsing are described in Appendix C.

6.2GetBackupListRequest Browser Frame

The GetBackupListRequest frame is sent by a browser client to any Master Browser for a domain to allow the client to learn the identities of Backup Browsers. To get the list of Backup Browsers for domain "D"; from the Local Master Browser for that domain, the GetBackupListRequest browser frame is sent ~~on mailslot “\MAILSLOT\MSBROWSE” and to NETBIOS unique names D(1d) and D(1b). (The frame is sent simultaneously t~~to to NETBIOS unique name D(1d) and mailslot “\MAILSLOT\MSBROWSE”. To ~~both NETBIOS names; the first response is used.)~~get the list of Backup Browsers for domain "D" from the Domain Master Browser for that domain, the GetBackupListRequest browser frame is sent to to NETBIOS unique name D(1b) and mailslot “\MAILSLOT\MSBROWSE”. The definition of the GetBackupListRequest frame is:

```
struct {
     unsigned ~~short~~char OpCode;
     unsigned short Token;
}
```

where :

OpCode identifies this structure as a request to return a list of Backup servers. Opcode is defined as GetBackupListRequest and has a value of decimal 9.

Token is a handle of meaning only to the client issuing the browser frame. The Local Master Browser will return this token unmodified in the response. The client should use this to distinguish replies to multiple outstanding GetBackupList requests. This implies that every GetBackupListRequest should have an unique handle, at least within the outstanding lifetime of a request.

The expected response is a GetBackupListResponse frame (see next section).

6.3GetBackupListResponse Browser Frame

The GetBackupListResponse frame is sent by a Master Browser in response to a GetBackupListRequest frame. ~~If the GetBackupListRequest was sent from the computer whose name is "ComputerName", t~~The GetBackupListResponse frame is sent to the ~~ComputerName(00)~~ NETBIOS unique name in the SOURCE_NAME of the mailslot message containing the GetBackupListRequest and mailslot \MAILSLOT\LANMAN. Note: this name is not part of the ~~request and the Master~~body of the request and on many systems the ~~Browser needs to “deduce” this name with some cooperation from the transport protocol~~

~~involved~~Master Browser will need to obtain this name from the NetBIOS service. The definition of the GetBackupListResponse frame is:

```
struct {
     unsigned ~~short~~char OpCode;
     unsigned short BackupServerCount;
     unsigned short Token;
     unsigned char  BackupServerList[][]
}
```

where:

Opcode ——Identifies this structure as a backup list.

BackupServerCount ——Specifies the number of backup servers that follow this list.

Token ——Is returned unmodified to the client. This is used by the client to associate an incoming **BackupListResponse** with its **BackupListRequest.**

BackupServerList ——ASCII backup servers. Each server name is null terminated and up to 16 bytes in length.

6.4The NetServerEnum2 RAP Service

The **NetServerEnum2** RAP service lists all computers of the specified type or types that are visible in the specified domains. It may also enumerate domains.

The following definition uses the notation and terminology defined in the CIFS Remote Administration Protocol specification, which is required in order to make it well-defined. The definition is:

```
unsigned short NetServerEnum2 (
     unsigned short sLevel,
     RCVBUF         pbBuffer,
     RCVBUFLEN      cbBuffer,
     ENTCOUNT       pcEntriesRead,
     unsigned short *pcTotalAvail,
     unsigned long  fServerType,
     char           *pszDomain,
);
```

where:

sLevel specifies the level of detail (0 or 1) requested.

pbBuffer points to the buffer to receive the returned data. If the function is successful, the buffer contains a sequence of **server_info_***x* structures, where *x* is 0 or 1, depending on the level of detail requested.

cbBuffer specifies the size, in bytes, of the buffer pointed to by the *pbBuffer* parameter.

pcEntriesRead points to a 16 bit variable that receives a count of the number of servers enumerated in the buffer. This count is valid only if **NetServerEnum2** returns the NERR_Success or ERROR_MORE_DATA values.

pcTotal Avail points to a 16 bit variable that receives a count of the total number of available entries. This count is valid only if **NetServerEnum2** returns the NERR_Success or ERROR_MORE_DATA values.

fServerType specifies the type or types of computers to enumerate. Computers that match at least one of the specified types are returned in the buffer. Possible values are defined in the request parameters section.

pszDomain points to a null-terminated string that contains the name of the workgroup in which to enumerate computers of the specified type or types. If the *pszDomain* parameter is a null string or a null pointer, servers are enumerated for the current domain of the computer.

6.4.1Transaction Request Parameters section

The Transaction request parameters section in this instance contains:

- The 16 bit function number for NetServerEnum2 which is 104.
- The parameter descriptor string which is "WrLehDz".
- The data descriptor string for the (returned) data which is "B16" for level detail 0 or "B16BBDz" for level detail 1.
- The actual parameters as described by the parameter descriptor string.

The parameters are:

- A 16 bit integer with a value of 0 or 1 (corresponding to the "W" in the parameter descriptor string. This represents the level of detail the server is expected to return
- A 16 bit integer that contains the size of the receive buffer.
- A 32 bit integer that represents the type of servers the function should enumerate. The possible values may be any of the following or a combination of the following:

SV_TYPE_WORKSTATION	0x00000001	All workstations
SV_TYPE_SERVER	0x00000002	All servers
SV_TYPE_SQLSERVER	0x00000004	Any server running with SQL server
SV_TYPE_DOMAIN_CTRL	0x00000008	Primary domain controller
SV_TYPE_DOMAIN_BAKCTRL	0x00000010	Backup domain controller
SV_TYPE_TIME_SOURCE	0x00000020	Server running the timesource service
SV_TYPE_AFP	0x00000040	Apple File Protocol servers
SV_TYPE_NOVELL	0x00000080	Novell servers
SV_TYPE_DOMAIN_MEMBER	0x00000100	Domain Member
SV_TYPE_PRINTQ_SERVER	0x00000200	Server sharing print queue
SV_TYPE_DIALIN_SERVER	0x00000400	Server running dialin service.
SV_TYPE_XENIX_SERVER	0x00000800	Xenix server
SV_TYPE_NT	0x00001000	NT server
SV_TYPE_WFW	0x00002000	Server running Windows for Workgroups
SV_TYPE_SERVER_NT	0x00008000	Windows NT non DC server
SV_TYPE_POTENTIAL_BROWSER	0x00010000	Server that can run the browser service
SV_TYPE_BACKUP_BROWSER	0x00020000	Backup browser server
SV_TYPE_MASTER_BROWSER	0x00040000	Master browser server
SV_TYPE_DOMAIN_MASTER	0x00080000	Domain Master Browser server
SV_TYPE_LOCAL_LIST_ONLY	0x40000000	Enumerate only entries marked "local"local entries (marked "authoritative"). This is meaningful only for the NetServerEnum2 request and should be ignored within the NetServerEnum2 response.
~~SV_TYPE_LOCAL_LIST_ONLY~~	~~0x40000000~~	~~Enumerate only entries marked "local"~~
SV_TYPE_DOMAIN_ENUM	0x80000000	Enumerate Domains. The pszDomain parameter must be NULL.

- A null terminated ASCII string representing the pszDomain parameter described above

6.4.2Transaction Request Data section

There is no data or auxiliary data to send as part of the request.

6.4.3Transaction Response Parameters section

The transaction response parameters section consists of:

- A 16 bit word indicating the return status. The possible values are:

Code	Value	Description
NERR_Success	0	No errors encountered
ERROR_MORE_DATA	234	Additional data is available
NERR_ServerNotStarted	2114	The RAP service on the remote computer is not running
NERR_BadTransactConfig	2141	The server is not configured for transactions, IPC$ is not shared

- A 16 bit "converter" word.
- A 16 bit number representing the number of entries returned.
- A 16 bit number representing the total number of available entries. If the supplied buffer is large enough, this will equal the number of entries returned.

6.4.4Transaction Response Data section

The return data section consists of a number of SHARE_INFO_1 structures. The number of such structures present is determined by the third entry (described above) in the return parameters section.

At level detail 0, the Transaction response data section contains a number of SERVER_INFO_0 data structure. The number of such structures is equal to the 16 bit number returned by the server in the third parameter in the Transaction response parameter section. The SERVER_INFO_0 data structure is defined as:

```
struct SERVER_INFO_0 {
     char      sv0_name[16];
};
```

where:

sv0_name is a null-terminated string that specifies the name of a computer or domain .

At level detail 1, the Transaction response data section contains a number of SERVER_INFO_1 data structure. The number of such structures is equal to the 16 bit number returned by the server in the third parameter in the Transaction response parameter section. The SERVER_INFO_1 data structure is defined as:

```
struct SERVER_INFO_1 {
     char           sv1_name[16];
     char           sv1_version_major;
     char           sv1_version_minor;
     unsigned long  sv1_type;
     char           *sv1_comment_or_master_browser;
};
```

sv1_name contains a null-terminated string that specifies the name of a computer, or a domain name if SV_TYPE_DOMAIN_ENUM is set in sv1_type.

sv1_version_major whatever was specified in the HostAnnouncement or DomainAnnouncement frame with which the entry was registered.

sv1_version_minor whatever was specified in the HostAnnouncement or DomainAnnouncement frame with which the entry was registered.

sv1_type specifies the type of software the computer is running. The member can be one or a combination of the values defined above in the Transaction request parameters section for fServerType.

sv1_comment_or_master_browser points to a null-terminated string. If the sv1_type indicates that the entry is for a domain, this specifies the name of server running the domain master browser; otherwise, it specifies a comment describing the server. The comment can be a null string or the pointer may be a null pointer.

In case there are multiple SERVER_INFO_1 data structures to return, the server may put all these fixed length structures in the return buffer, leave some space and then put all the variable length data (the actual value of the sv1_comment strings) at the end of the buffer.

There is no auxiliary data to receive.

6.5HostAnnouncement Browser Frame

To advertise its presence, i.e. to publish itself as being available, a non-browser server sends a HostAnnouncement browser frame. If the server is a member of domain "D", this frame is sent to the NETBIOS unique name D(1d) and mailslot "\MAILSLOT\MSBROWSE". The definition of the HostAnnouncement frame is:

```
struct {
    unsigned ~~short~~char Opcode;
    unsigned char UpdateCount;
    unsigned long Periodicity;
    unsigned char ServerName[];
    unsigned char VersionMajor;
    unsigned char VersionMinor;
    unsigned long Type;
    unsigned long Signature;
    unsigned char Comment[];
}
```

where:

Opcode ——Identifies this structure as a browser server announcement and is defined as HostAnnouncement with a value of decimal 1.

UpdateCount – must be sent as zero and ignored on receipt.

Periodicity ——The announcement frequency of the server (in milliseconds). The server will be removed from the browse list if it has not been heard from in 3X its announcement frequency. In no case will the server be removed from the browse list before the period 3X has elapsed. Actual implementations may take more than 3X to actually remove the server from the browse list.

ServerName ——Null terminated ASCII server name (up to 16 bytes in length). This name SHOULD be registered with NetBIOS by the server offering the services specified in the Type field.

VersionMajor ——The major version number of the OS the server is running. it will be returned by NetServerEnum2.

VersionMinor ——The minor version number of the OS the server is running. This is entirely informational and does not have any significance for the browsing protocol.

Type ——Specifies the type of the server. The server type bits are specified in the NetServerEnum2 section.

Signature —— The browser protocol minor version number in the low 8 bits, the browser protocol major version number in the next higher 8 bits and the signature `0xaa55` in the high 16 bits of this field. Thus, for this version of the browser protocol (1.15) this field has the value `0xaa55010f`. This may used to isolate browser servers that are running out of revision browser software; otherwise, it is ignored.

Comment ——Null terminated ASCII comment for the server. Limited to 43 bytes.

When a non-browser server starts up, it announces itself in the manner described once every minute. The frequency of these statements is gradually stretched to once every 12 minutes.

Note: older non-browser servers in a domain "D" sent HostAnnouncement frames to the NETBIOS group name D(00). Non-Browser servers supporting version 1.15 of the browsing protocol SHOULD NOT use this NETBIOS name, but for backwards compatibility Master Browsers MAY receive and process HostAnnouncement frames on this name as described above for D(1d).

6.6AnnouncementRequest Browser Frame

When a Master Browser starts up and its browse list is empty, it may force all servers to announce themselves by broadcasting an AnnouncementRequest frame. If the Master Browser serves domain "D", the AnnouncementRequest frame is broadcast using the NETBIOS group name D(00) and mailslot “\MAILSLOT\LANMAN”. The definition of the AnnouncementRequest frame is:

```
struct {
     unsigned ~~short~~char Opcode;
     unsigned char  ResponseComputerName[];
};
```

Opcode ——Identifies this structure as an announcement request and is defined as AnnounceMent Request with a value of decimal 2.

ResponseComputerName ——Specifies the name of the computer to send the server announcement to and is up to 16 bytes in length. This is ignored . The response to this browser frame is a HostAnnouncement browser frame as described immediately above. That browser frame does not use this parameter at all.

Recipients of this packet should reply by sending an HostAnnouncement frame as described above. The reply should be sent within a randomly determined time period that may have a duration of up to 30 seconds. The random delay ensures that the Master Browser who sent out the packet does not get flooded with replies, all at the same time.

6.7RequestElection Browser Frame

To force the election of a new Master Browser for a domain, any browser client or server can broadcast a RequestElection frame. If the election is for domain "D", the frame is broadcast using the NETBIOS group name D(1e) and mailslot “\MAILSLOT\MSBROWSE”. The definition of the RequestElection frame is:

```
struct {
    unsigned ~~short~~char Opcode;
    unsigned char Version;
    unsigned long Criteria;
    unsigned long TimeUp;
    unsigned long MustBeZero;
    unsigned char ServerName[];
}
```

Opcode ——Identifies this structure as an election request, and is defined as RequestElection, with a value of decimal 8.

Version ——Specifies the version of this election packet. This is a constant and always has the value 0x00010f00

Criteria ——Specifies the election criteria of the sender. Produced by OR'ing together the *Version* and the following:

```
OS info:
     Windows for Workgroups & Windows 95:   0x00000000
     Windows NT:                            0x01000000
     Windows NT Server:                     0x02000000
Role:
     PDC:                                   0x00000080
     Preferred Master:                      0x00000008
     Running Master:                        0x00000004
     Backup Browser which was
     recently a Master Browser:             0x00000002
     Running Backup Browser:                0x00000001
     Using NBNS for NETBIOS:                0x00000020
```

The following masks can be used to isolate parts of the *Criteria*:

```
Operating System Type Mask                  0xFF000000
Election Protocol Version Mask:             0x00FFFF00
Per version criteria mask:                  0x000000FF
```

TimeUp ——The number of seconds that the server has been up.

MustBeZero——Must be zero.

ServerName ——Null terminated ASCII server name (up to 16 bytes in length).

6.8Browser Elections

All browsers for a domain "D" MUST listen for RequestElection frames on the group name D(1e) and mailslot “\MAILSLOT\MSBROWSE”.

Elections proceed in rounds. A round is initiated when a RequestElection frame is sent. When a Browser receives a RequestElection frame, it determines if it has won the round using the following rules:

If it has lost an election in the last several seconds, it loses
If its election *Version* is greater than the senders election *Version*, it wins
Else if its election *Criteria* (including the election version) is greater than the senders *Criteria*, it wins

Else if it has been up longer than the sender, it wins
Else if its name is lexically lower than the sender's name, it wins
(I.e., at this point, a sever named A will become Master Browser over a server named X)

(Note that many browsers which receive a RequestElection frame may win a round.)

Each time it wins a round, a browser sends out a RequestElection frame, after a delay based on the browser's current role in the domain:

- Master Browsers and Domain Master Browsers delay for 100 ms.
- Backup Browsers delay for an amount randomly chosen from the interval 200-600 ms.
- All other browsers delay for an amount randomly chosen from the interval 800-3000 ms.

If a browser loses a round it drops out of the election by ignoring RequestElection frames until it receives a LocalMasterAnnouncement frame that tells which system is the new Master Browser.

If a browser wins 4 rounds in a row, it becomes the Master Browser.

6.9BecomeBackup Browser Frame

If a Local Master Browser for a domain "D" wants to promote a Potential Browser to Backup Browser, it broadcasts a BecomeBackup frame using the NETBIOS group name D(1e) and the “\MAILSLOT\MSBROWSE” mailslot. The definition of the BecomeBackup frame is:

```
struct {
     unsigned ~~short~~char Opcode;
     unsigned char BrowserToPromote[];
}
```

Opcode —— Identifies this structure as a browser server announcement, is defined as BecomeBackup, with a value of decimal 11

BrowserToPromote ——Specifies the name of the browser server to be promoted to backup. Maximum of 16 bytes in length.

6.10LocalMasterAnnouncement Browser Frame

A Local Master Browser for a domain announces itself to all the other browsers in its domain that are on its subnet using the LocalMasterAnnouncement frame. If the Local Master Browser serves domain "D", the LocalMasterAnnouncement frame is broadcast using the NETBIOS group name D(1e) and the mailslot “\MAILSLOT\MSBROWSE”. The definition of the LocalMasterAnnouncement frame is:

```
struct {
    unsigned shortchar Opcode;
    unsigned char UpdateCount;
    unsigned long Periodicity;
    unsigned char ServerName[];
    unsigned char VersionMajor;
    unsigned char VersionMinor;
    unsigned long Type;
    unsigned long Signature;
    unsigned char Comment[];
}
```

where:

Opcode ——Identifies this structure as a browser server announcement and is defined as LocalMasterAnnouncement with a value of decimal 15.

UpdateCount – must be sent as zero and ignored on receipt.

Periodicity ——The announcement frequency of the browser (in milliseconds). The browser will be removed from the browse list if it has not been heard from in 3X its announcement frequency. In no case will the server be removed from the browse list before the period 3X has elapsed. Actual implementations may take more than 3X to remove the server from the browse list.

ServerName ——Null terminated ASCII server name (up to 16 bytes in length).

VersionMajor ——The major version of the OS the server is running. This value is informational and irrelevant to the browsing protocol.

VersionMinor ——The minor version of the OS the server is running. This value is informational and irrelevant to the browsing protocol.

Type ——Specifies the type of the browser. The type bits are specified in the description of NetServerEnum2.

Signature —— The browser protocol minor version number in the low 8 bits, the browser protocol major version number in the next higher 8 bits and the signature `0xaa55` in the high 16 bits of this field. This may used to isolate browser servers that are running out of revision browser software; otherwise, it is ignored. Thus, for this version of the browser protocol (1.15) this field has the value `0xaa55010f`.

Comment ——Null terminated ASCII comment for the browser. Limited to 43 bytes.

Local Master Browsers do not need to send HostAnnouncement frames; the LocalMasterAnnouncement accomplishes that function.

6.11MasterAnnouncement browser Frame

The MasterAnnouncement frame is sent by a Local Master Browser to the Domain Master Browser, which runs on the PDC. If the name of the PDC is "PDCName", then the MasterAnnouncement frame is sent to the NETBIOS unique name PDCName(00) and mailslot “\MAILSLOT\MSBROWSE”. Appendix B describes how to determine the name of the Primary Domain Controller. The definition of the MasterAnnouncement frame is::

```
struct {
     unsigned ~~short~~ char Opcode;
     unsigned char  MasterBrowserServerName[];
};
```

Opcode ——Identifies this structure as a master browser server announcement and is defined as MasterAnnouncement with a value of decimal 13.

MasterBrowserServerName ——Specifies the name of the master browser server (up to 16 bytes in length).

6.12 DomainAnnouncement Browser Frame

Master Browsers (including Local Master Browsers and Domain Master Browsers) announce the domain they serve to any other Master Browsers on their subnet by broadcasting a DomainAnnouncement frame using the NETBIOS group name "(01)(02)_~~MSBROWSE~~_MSBROWSE_ _(02)(01)" and mailslot "\MAILSLOT\MSBROWSE". The definition of the DomainAnnouncement frame is:

```
struct {
     unsigned ~~short~~ char Opcode;
     unsigned char UpdateCount;
     unsigned long Periodicity;
     unsigned char DomainName[];
     unsigned char VersionMajor;
     unsigned char VersionMinor;
     unsigned long Type;
     unsigned long Signature;
     unsigned char MasterBrowserName[];
}
```

where:

Opcode ——Identifies this structure as a browser server announcement and is defined as DomainAnnouncement with a value of decimal 12.

UpdateCount – must be sent as zero and ignored on receipt.

Periodicity ——The announcement frequency of the domain (in milliseconds). The domain will be removed from the browse list if it has not been heard from in 3X its announcement frequency. In no case will the domain be removed from the browse list before the period 3X has elapsed. Actual implementations may take more than 3X to actually remove the domain from the browse list.

DomainName ——Null terminated ASCII server name (up to 16 bytes in length).

VersionMajor ——The major version of the OS the server is running. This value is informational and irrelevant to the browsing protocol.

VersionMinor ——The minor version of the OS the server is running. This value is informational and irrelevant to the browsing protocol.

Type ——Specifies the type of the server. The server type bits are specified in the previous section.

Signature —— The browser protocol minor version number in the low 8 bits, the browser protocol major version number in the next higher 8 bits and the signature `0xaa55` in the high 16 bits of this field. This may used to isolate browser servers that are running out of revision browser software; otherwise,

it is ignored. Thus, for this version of the browser protocol (1.15) this field has the value `0xaa55010f`.

MasterBrowserName ——Null terminated ASCII string containing the name of the master browser server for this domain.

7References

[CIFS 96} I. Heizer, P. Leach, D. Perry, "Common Internet Files System Protocol (CIFS/1.0)", Internet-Draft, <draft-heizer-cifs-v1-spec-00.txt>,June 30, 1996 . (Work in Progress)

[RFC 1001] K. Auerbach, A. Aggarwal, "Protocol Standard for a NETBIOS Service on a TCP/UDP Transport: Concepts and Methods", RFC 1001, Epilogue Technology, March 1987.

[Bradner 96] S. Bradner, ""Key words for use in RFCs to Indicate Requirement Levels", Internet-Draft, <draft-bradner-key-words-02.txt>, August 1996 (Work in Progress)

8Author's Addresses

Paul Leach
Dilip Naik
Microsoft
1 Microsoft Way
Redmond, WA 98052
paulle@microsoft.com
v-dilipn@microsoft.com

9Appendix A - Multi-net considerations

To begin with, let’s clearly define what is meant by multiple networks here. A computer can be running one or more network protocols on a single network adapter card such as IP and IPX. Each of these is considered to be a “network” for the purposes of this paragraph. A computer could also have multiple network adapter cards, and there could be different protocols running on the different adapter cards, or even the same protocol(s) on all of the adapter cards. So, more precisely, a network here means a transport protocol per adapter card. A computer with 2 physical network cards, running 2 different transport protocols on each network card, would have 4 logical networks.

Browsers need to remember which logical network a server is located on. When a client queries a browser for a list of servers, the browser server needs to return a list of servers that are on the same logical network on which the client query arrived at the browser server. So a client that sends a browser frame using say IP will only be returned information about servers that sent announcements using IP.
To summarize, browser servers need to understand the concept of logical networks and track server announcements as well as client queries on a per logical network basis.

10Appendix B - Primary Domain Controller Location Protocol

This appendix details how a client goes about locating a Primary Domain Controller (PDC). The process is rather involved, because different versions of the PDC have used different versions of the protocol, and hence a client that does not know what protocol is supported by its PDC has to try them all.

A Primary Domain Controller (PDC) for a domain "D" is located by sending a mailslot message containing a NETLOGON_QUERY frame to a NETBIOS name and mailslot "\NET\NETLOGON" and then waiting for a reply mailslot message, which will be sent to the mailslot name specified by the client in the NETLOGON_QUERY structure., and which will contain a NETLOGON_RESPONSE structure. If there is no response after a delay, the message may be retransmitted. The delay MUST be at least twice the expected service time, and the delay should be doubled after each time-out.

If a reply is received, the name of the PDC SHOULD be cached for future use, so as time minimize network traffic. If no reply is received after several retransmissions, the PDC may be declared to be unreachable, and no further attempt to locate it should be made for a while (exactly how long depends on the expected recovery time for a PDC and/or for the network; typically a minute or so, but should be increased after each failure).

The only difference between versions of the protocol is the NETBIOS name to which the message is sent, as follows:

NETBIOS name	name type	PDC's OS version
D(1b)	unique	Windows NT 3.51 or later or compatible
D(1c)	group	Windows NT 3.1 or later or compatible
D(00)	group	all

Clients which are configured to know or are willing to assume what version of the protocol their PDC is running may directly use the appropriate NETBIOS name for that version. Otherwise, they SHOULD first attempt D(1b), since it is unicast and creates the least network traffic; if there is no response, then they SHOULD try the others. They MAY try them in parallel.

The NETLOGON_QUERY structure is defined as :

```
struct NETLOGON_QUERY{
     unsigned short charOpcode;
     char               ComputerName[];
     char               MailslotName[];
     unsigned short     Lm20Token;
} ;
```

Opcode ——Identifies this structure as a NETLOGON_QUERY and has a value of 0x07.

ComputerName ——Specifies the ASCII name of the computer sending the query, and is up to 16 bytes in length. The response is sent to NETBIOS unique name <ComputerName>(00).

MailslotName ——Specifies the ASCII name of the mailslot to which the response is to be sent, and is up to 256 bytes in length; cannot be “\MAILSLOT\LANMAN” or “\MAILSLOT\MSBROWSE” or "\NET\NETLOGON".

Lm20Token - has a value of 0xFFFF.

The response mailslot message contains a NETLOGON_RESPONSE data structure that is defined as the following for non Windows NT clients:

```
struct NETLOGON_RESPONSE
{
     unsigned short charOpcode;
     char               PrimaryDCName[16];
     unsigned short     Lm20Token;
};
```

where

Opcode ——Identifies this structure as a NETLOGON_RESPONSE and has a value of 0x12.

PrimaryDCName ——Specifies the ASCII name of the Primary Domain Controller and is up to 16 bytes in length.

Lm20Token - has a value of 0xFFFF

Note that this procedure to locate a Primary Domain Controller is expensive in terms of network traffic. The Microsoft implementations attempt to alleviate this by caching the PDC Name. Before using the cached PDC Name, a NetServerEnum2 API is remoted to the PDC and a sanity check is performed to ensure that the server type returned indicates a Primary Domain Controller

11Appendix C - Summary of Special NETBIOS Names

This section details the various NETBIOS names that are involved in sending and receiving browser frames. The different mailslots involved in browsing (there are only 3) are described later on when the browser frames are detailed.

11.1Registered unique names

<COMPUTER>(00)

This name is used by all servers and clients to receive second class mailslot ~~request~~messages. A system must add this name in order to receive mailslot messages. The only browser requests that should appear on this name are *BecomeBackup*, *GetBackupListResp*, *MasterAnnouncement,* and *LocalMasterAnnouncement* frames. All other datagrams (other than the expected non-browser datagrams) may be ignored and an error logged.

<DOMAIN>(1d)

This name is used to identify a master browser server for domain "DOMAIN" on a subnet. A master browser server adds this name as a unique NETBIOS name when it ~~comes up~~becomes master browser. If the attempt to add the name fails, the master browser server assumes that there is another master in the domain and will fail to come up. It may log an error if the failure occurs more than 3 times in a row (this either indicates some form of network misconfiguration or a software error). The only requests that should appear on this name are *GetBackupListRequest* and *HostAnnouncement* requests. All other datagrams on this name may be ignored (and an error logged). If running a NETBIOS name service (NBNS, such as WINS), this name should not be registered with the NBNS.

<DOMAIN>(1b)

This name is used to identify the Domain Master Browser for domain "DOMAIN" (which is also the primary domain controller). It is a unique name added only by the primary domain controller. The primary domain controller will respond to *GetBackupListRequest* on this name just as it responds to these requests on the <DOMAIN>(1d) name.

11.2Registered group names

(01)(02)~~__MSBROWSE__~~__MSBROWSE__(02)(01)

This name is used by Master Browsers to announce themselves to the other Master Browsers on a subnet. It is added as a group name by all ~~masters b~~Master Browser servers. The only broadcasts that should appear on this name is *DomainAnnouncement* requests. All other datagrams can be ignored.

<DOMAIN>(00)

This name is used by clients and servers in domain "DOMAIN" to process server announcements. The only requests that should appear on this name that the browser is interested in are *AnnouncementRequest* and NETLOGON_QUERY (to locate the PDC) packets. All other unidentifiable requests may be ignored (and an error logged).

<DOMAIN>(1e)

This name is used for announcements to browsers for domain "DOMAIN" on a subnet. This name is registered by all

the browser servers in the domain. The only requests that should appear on this name are *RequestElection* and *AnnouncementRequest* packets. All other datagrams may be ignored (and an error logged).

`<DOMAIN>(1c)`
: This name is registered by Primary Domain Controllers.

12Appendix D - Browsing Protocol Evolution

This Appendix details how the Microsoft Browser specification evolved and correlates the evolution to specific Microsoft products.

The first browser implementation was with Lan Manager 1.0. Here, there were no browser servers as such. Each client acted as its own browser server. All servers announced themselves by means of datagrams and every client and server listened for those datagrams. Obviously, the amount of browser datagram traffic is fairly high and scales extremely poorly with an increase in the number of servers.

The next major revision in the browser specification was with Windows For Workgroups. This is where the concept of a browser server was really introduced.

Windows NT 3.51 expanded upon what Windows For Workgroups built. Windows NT 3.51 introduced the special NETBIOS name <Domain>(1b) that is registered with WINS. Windows NT 3.51 also shipped a new redirector for Windows For Workgroups that could take advantage of this new <Domain>(1b) name.

In a domain which spans multiple broadcast areas, it may be necessary to have a configuration file available that can resolve the address of a browser server. This is because browsers rely on broadcasts for name resolution, for historical reasons. But these name resolution broadcast packets are not forwarded by routers that span the multiple broadcast areas of a domain. One example of such a configuration file is the LMHOSTS file on Windows NT machines.