[MS-BRWS]:
Common Internet File System (CIFS) Browser Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
04/03/2007	1.0		MCPP Milestone Longhorn Initial Availability
07/03/2007	2.0	Major	MLonghorn+90
07/20/2007	2.0.1	Editorial	Revised and edited the technical content.
08/10/2007	3.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
09/28/2007	4.0	Major	Updated and revised the technical content.
10/23/2007	4.0.1	Editorial	Revised and edited the technical content.
11/30/2007	4.1	Minor	Revised links.
01/25/2008	4.1.1	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction

This document is a specification of the Common Internet File System (CIFS) Browser Protocol (version 1.10).

The CIFS Browser Protocol defines the messages that are sent and received by a server that acts as a clearinghouse for services available on the network, servers that are making services such as printing or file sharing available on the network, and clients requesting the details of a particular service.

1.1 Glossary

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

Active Directory (AD)
ASCII
Browser
Browser Server
Domain
Little-Endian
Mailslot
NetBIOS Name
Primary Domain Controller (PDC)

The following terms are specific to this document:

Backup Browser Server: A **browser server** that was selected by the **local master browser server** on that **subnet** to be available to share the processing load that is required to serve **browser clients**, as specified in section 3.1. **Backup browser servers** keep copies of the information that is maintained by the **local master browser server** by periodically querying that server.

Browser Client: A computer on the network that queries or sends information to a **browser server**. There are three types of **browser clients**: workstations, **nonbrowser servers**, and **browser servers**. In the context of browsing, **nonbrowser servers** supply information about themselves to **browser servers**, and workstations query **browser servers** for information. **Browser servers** can behave as clients and query other **browser servers**.

Election Criteria: The collective information in a **browser** RequestElection (section 2.2.4) packet that is used to determine the winner of an election.

Frame: A CIFS Browser Protocol message.

Group Name: A 16-byte formatted NetBIOS computer name, which can have multiple IP addresses assigned to it. That is, multiple NetBIOS nodes (processor locations) can use this name to register for services, as specified in [RFC1001].

Machine Group: A generic reference to a **domain** or a **workgroup**, of which a specified machine is a member. A computer implementing the CIFS Browser Protocol must be a member of either a **workgroup** or a **domain**.

Nonbrowser Server: A server that wants to be enumerated to clients of the CIFS Browser Protocol that does not otherwise implement elements of the CIFS Browser Protocol.

Platform: A specific operating system that is a standard for the development and operation of computers.

Potential Browser Server: A **browser server** that is capable of being a **backup browser server** or a **master browser server** but is not currently fulfilling either role.

Preferred Master Browser Server: A machine that functions as a typical **backup browser server** except that it forces a **browser** election when it is started. Preferred master browser servers are given an advantage in elections. By configuring one or more machines as preferred master browser servers, a network administrator can actually choose particular machines for this role.

Subnet: A logical division of a network. Subnets provide a multilevel hierarchical routing structure for the Internet. On TCP/IP networks, subnets are defined as all devices whose IP addresses have the same prefix. Subnets are useful for both security and performance reasons. In general, broadcast messages are scoped to within a single subnet. For more information about subnets, see [RFC1812].

Unique Name: A 16-byte formatted NetBIOS computer name that can have only one IP address assigned to it; that is, only a single NetBIOS node (or processing location) can use this name to register for services, as specified in [RFC1001].

Windows Internet Name Service (WINS): A name service for the NetBIOS protocol that is specifically designed to ease transition to a TCP/IP–based network.

Workgroup: A collection of computers that share a name. In the absence of a **domain**, a workgroup allows a convenient means for **browser clients** to limit the scope of a search.

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.

[MS-BRWSA] Microsoft Corporation, "Common Internet File System (CIFS) Browser Auxiliary Protocol Specification", September 2007.

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

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

[MS-MAIL] Microsoft Corporation, "Remote Mailslot Protocol Specification", March 2007.

[MS-RAP] Microsoft Corporation, "Remote Administration Protocol Specification", March 2007.

[MS-SMB] Microsoft Corporation, "Server Message Block (SMB) Protocol Specification", July 2007.

[MS-SRVS] Microsoft Corporation, "Server Service Remote Protocol Specification", January 2007.

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

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

1.2.2 Informative References

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification", June 2007.

[RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, June 1995, http://www.ietf.org/rfc/rfc1812.txt

1.3 Protocol Overview (Synopsis)

The CIFS Browser Protocol makes it possible for the following:

- A server (or a set of servers) to act as a clearinghouse (or **browser server**) for information about the services available in the network.
- A set of servers (**nonbrowser servers**) that are making services available to access the clearinghouse and advertise the services they offer.
- A set of clients (**browser clients**) to access the information clearinghouse and seek details of a particular service.

The two main objectives of the CIFS Browser Protocol are to:

- Update all **backup browser servers** with the contents of the response every time a browser client successfully receives a response to a **NetServerEnum2** request, as specified in [MS-RAP] section 2.5.5.2.
- Share the processing load of enumerating the services available in the network across different servers.

In the context of the CIFS Browser Protocol, browsing is a process for discovering servers that offer particular services. To provide those services, browser servers may assume different roles in their lifetimes and can dynamically switch between these roles.

Clients of browser servers are of three types:

- Workstations, which query browser servers for the information they contain.
- Nonbrowser servers, which supply browser servers with information by registering with them.
- And browser servers, which may behave as clients and query other browser servers.

The CIFS Browser Protocol manages groups of computers. This document refers to such a group of computers as a **machine group**. Machine groups provide a convenient means for clients to restrict the scope of a search when they query browser servers.

A machine group can be either a **workgroup** or a **domain**. In a workgroup configuration, browsing is limited to the scope of a single **subnet**. If computers are arranged in a Windows NT security domain, the CIFS Browser Protocol allows for browsing across multiple subnets. This functionality is enabled by a special browser server role that is known as the **domain master browser server**. This role is usually the responsibility of the **primary domain controller (PDC)**, which manages user access and security in the domain.

One browser server for each machine group on a subnet is selected as the **local master browser server** for that machine group. The selection occurs by an election process, as specified in sections

2.2.4 and 3.3.6. Servers that are in the local master browser server machine group on the subnet register with it, as do the local master browser server for other machine groups on the subnet. The local master browser server uses these registrations to maintain authoritative information about its machine group on its subnet. If there are servers in the domain that are located on other subnets on the network, the local master browser server for the domain can obtain information about them from the domain master browser server of the domain.

A backup browser server on a subnet is a browser server that was selected by the local master browser server on that subnet to be available to share the processing load that is required to serve browser clients, as specified in section 3.1. Backup browser servers keep copies of the information that is maintained by the local master browser server by periodically querying that server.

Multiple backup browser servers may exist on a subnet; the number of local master browser servers is typically configured to enable enough backup browser servers to handle the expected query load. Clients can find backup browser servers by querying the local master browser server. Clients on a subnet query backup browser servers on the subnet, not the local master browser server; also, they are expected to spread their queries evenly across backup browser servers to balance the load.

When a domain spans multiple subnets, the domain master browser server is responsible for keeping them synchronized. The domain master browser server periodically queries all the local master browser servers for lists that contain all the known domains (and all the servers in their domains) within their subnets. The domain master browser server merges all the replies into a single master list, which allows it to act as a collection point for inter-subnet browsing information. The local master browser servers periodically query the domain master browser server to retrieve the network-wide information it maintains.

When a domain spans only a single subnet, there is not a distinct local master browser server; this role is instead handled by the domain master browser server. Similarly, the domain master browser server is always the local master browser server for its domain on its own subnet.

When a browser client suspects that the local master browser server for its machine group has failed, the client initiates an election process in which the browser servers participate. This election process is specified in sections 2.2.4 and 3.3.6. When this election process occurs, some browser servers may change roles.

1.4 Relationship to Other Protocols

The CIFS Browser Protocol depends on the following protocols:

- **Mailslots**, as specified in [MS-MAIL].
- Remote Administration Protocol (RAP), as specified in [MS-RAP].
- Browser-specific **mailslot** messages sent as payloads of datagrams (as specified in [RFC1001] section 17). These datagrams are sent to special NetBIOS names using NetBIOS as a transport.
- CIFS Transaction Server Messenger Block (SMB) data structure, as specified in [MS-SMB].
- CIFS/SMB messages and the associated transport mechanism, as specified in [MS-SMB].

1.5 Prerequisites/Preconditions

The CIFS Browser Protocol has the following preconditions.

- The Server Message Block (SMB) Protocol dialect that is negotiated between a browser client and server MUST be LANMAN1.0 or later, as specified in [MS-SMB] section 3.2.4.2.2.

- A NetBIOS–based datagram service implementation must be available.
- All primary domain controllers (PDCs) must implement the CIFS Browser Protocol, except on networks that are based exclusively on **Active Directory (AD)** that do not use the CIFS Browser Protocol, as specified in section 1.6. If the PDC for a domain does not implement the CIFS Browser Protocol, browser clients are not able to retrieve information about servers on subnets other than their own.

1.6 Applicability Statement

The CIFS Browser Protocol is used when automatic discovery of services offered within a network is expected, NetBIOS is available, and the network is not based exclusively on Active Directory. If all of the services offered on a network are specified within Active Directory, and all clients are capable of interrogating Active Directory for these services, there is no need for servers on that network to support the browser protocol. However, if all services available within a network are not specified in Active Directory, it is required for servers to support the CIFS Browser Protocol in order to discover the services that are offered in a network.

Fine-grained search criteria (that is, by location or by another attribute of a resource) are not supported by the CIFS browser protocol, so it is not scalable to servers that provide similar services. It is also not extensible to new service types beyond those specified herein, so the protocol is not suitable for discovering such services. Also, the browser protocol includes no security mechanism, and thus is not suitable to networking environments requiring secure discovery. In addition, all of the text elements implemented in the browser protocol are implemented as ASCII text, and thus are not suited to internationalization.

Finally, the information in the list of servers that can be returned by this protocol must fit in 64K bytes of data. This limits the number of systems that can be in a server list in a single machine group.

1.7 Versioning and Capability Negotiation

The CIFS Browser Protocol provides for a version field, as specified in section 2.2.4. It also specifies a biased election mechanism to nominate some servers as local master browser servers. This election mechanism, specified in section 3.3.6, is biased in favor of servers implementing newer versions of the CIFS Browser Protocol.

1.8 Vendor Extensible Fields

Some **frames** define **OSVersionMajor** and **OSVersionMinor** fields. These fields are returned to clients of the CIFS Browser Protocol. As such, implementations can use any values they want.<1>

This protocol uses Win32 error codes. These values are taken from the Windows error number space as specified in [MS-ERREF]. Vendors SHOULD reuse those values with their indicated meaning. Choosing any other value runs the risk of a collision in the future.<2>

1.9 Standards Assignments

The CIFS Browser Protocol uses the parameter assignments as shown in the following table.

Parameter	Value	Reference
Mailslot name	\MAILSLOT\LANMAN	As specified in [MS-MAIL]
Mailslot name	\MAILSLOT\BROWSE	As specified in [MS-MAIL]

For more information about NetBIOS naming conventions and control characters, see section 2.1.1.

2 Messages

CIFS Browser Protocol messages are mailslot messages, as specified in [MS-MAIL].

This document contains the following information on CIFS Browser Protocol messages.

- Section 2.1.1 specifies the recipients of CIFS Browser Protocol messages.
- Sections 2.2.1 through 2.2.10 specify the syntax of each CIFS Browser Protocol message.
- Section 3.3.5 specifies the details of CIFS Browser Protocol message processing, including events and sequencing rules.

2.1 Transport

The CIFS Browser Protocol depends on the Remote Mailslot Protocol transfer service, as specified in [MS-MAIL]. The CIFS Browser Protocol uses mailslot messages to accomplish inter-machine communications. This communication can be one-to-one or one-to-many. There are two specific names, \MAILSLOT\LANMAN and \MAILSLOT\BROWSE, that are used by the CIFS Browser Protocol. A browser server MUST accept **browser** requests on either of these mailslots. A browser client MAY select either mailslot to use for sending requests. Each message specifies the name it uses, as specified in section 2.2.<3>

The CIFS Browser Protocol also uses the SMB Protocol transport, as specified in [MS-SMB], which MUST be used to transport the SMB_COM_TRANSACTION request/response that contains in it the **NetServerEnum2** request, as specified in [MS-RAP] section 2.5.5.2.

2.1.1 NetBIOS Name Notation

The CIFS Browser Protocol encapsulates its messages in the Remote Mailslot Protocol, as specified in [MS-MAIL]. The Remote Mailslot Protocol requires a NetBIOS name for identification when specifying the origin of a mailslot message or the destination for a mailslot message. Additionally, CIFS Browser Protocol fields that require a NetBIOS name MUST be formatted as specified in [RFC1001] section 14. This section describes additional requirements when using NetBIOS names.

Before a NetBIOS name can be used, it MUST be registered with a name service as specified in [RFC1001] section 5.2.

NAME[<control byte>] denotes the ASCII string "NAME", which MUST be padded with spaces [0x20] to 15 bytes, with a signature value in the 16th byte. For example, the notation EXAMPLE[0x19] indicates a NetBIOS name that consists of the following hexadecimal bytes:

```
0x45, 0x58, 0x41, 0x4D, 0x50, 0x4C, 0x45, 0x20,
0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x19
```

Similarly, [<signature byte 1>][<signature byte 2>]NAME[<signature byte 3>][<signature byte 4>] denotes a NetBIOS name consisting of two signature bytes followed by the name and padded with spaces to a total of 14 bytes with the two additional signature bytes in the 15th and 16th bytes. The four signature bytes are not necessarily equal.

For example, the notation [01][02]__MSBROWSE__[02][01] indicates a NetBIOS name consisting of the hexadecimal bytes:

```
0x01, 0x02, 0x5F, 0x5F, 0x4D, 0x53, 0x42, 0x52,
0x4F, 0x57, 0x53, 0x45, 0x5F, 0x5F, 0x02, 0x01
```

Note There are two underscore (0x5F) characters before and after the word MSBROWSE. The meanings of specific signature bytes are listed in section 2.1.1.1.

Names that are placeholders and that need to be substituted with actual values are placed inside angle brackets (< >). Therefore, the string <domain> becomes REDMOND if the domain under consideration is named REDMOND. Details of the various NetBIOS names that are used for browsing are specified in the following sections.

2.1.1.1 Signature Bytes

NetBIOS signature bytes for computer and domain names are listed in the following table. Only the names related to the browser protocol are listed.

Name	Value	Name Type	Usage
<computer>	0x00	Unique	Workstation Service
[01][02]__MSBROWSE__[02][01]		Group	Master Browser
<machine group>	0x00	Group	Domain Name
<domain>	0x1B	Unique	Domain Master Browser
<machine group>	0x1D	Unique	Master Browser
<machine group>	0x1E	Group	Browser Service Elections

2.1.1.2 Unique Names

Name	Comment
<computer>[0x00]	This name is used by all servers and clients to receive second-class mailslot ([MS-MAIL] 3.2) messages. A system MUST register this NetBIOS name to receive mailslot messages. The only browser requests that use this name are GetBackupListResponse (section 2.2.6), MasterAnnouncement (section 2.2.9), and LocalMasterAnnouncement (section 2.2.10) frames. NetBIOS name registration is as specified in [RFC1001] section 5.2. Historical Note: The control code 0x00 was chosen because all computers implementing the CIFS Workstation service were (and still are) required to have that name registered with the NetBIOS name server; therefore, the name was guaranteed to be present on all computers. This is significant, because certain implementations of NetBIOS severely limited the number of NetBIOS names that could be registered on any given computer, and using an already existing name meant an additional name could be registered.
<machine group>[0x1D]	This name is used to identify a local master browser server for <machine group> on a subnet. A local master browser server MUST register this name as a NetBIOS **unique name** (as opposed to a **group name**.) The only requests that use this name are GetBackupListRequest (section 2.2.5), AnnouncementRequest (section 2.2.3), and HostAnnouncement (section 2.2.2) requests.<4>

Name	Comment
<domain>[0x1B]	This name MUST be added by the primary domain controller (PDC) as a unique name. All other servers MUST refrain from adding this name. This name is used to identify the domain master browser server for <domain>. A PDC responds to the GetBackupListRequest (section 2.2.5) request on this name.<5>

2.1.1.3 Group Names

Name	Comment
[0x01][0x02]__MSBROWSE__[0x02][0x01]	All local master browser servers MUST add this name as a group name. This name is used by local master browser servers to periodically announce themselves to local master browser servers for other domains on the subnet. The only message that uses this name is DomainAnnouncement (section 2.2.8).
<machine group>[0x00]	Browser clients and servers in <machine group> MUST register this name to process one-to-many mailslot messages. The only CIFS Browser Protocol message that uses this name is AnnouncementRequest (section 2.2.3).
<machine group>[0x1E]	All browser servers and potential browser servers within <machine group> MUST register this name to receive domain-wide broadcasts on a subnet. The only requests that use this name are RequestElection (section 2.2.4), BecomeBackup (section 2.2.7) and LocalMasterAnnouncement (section 2.2.10) frames.

2.2 Message Syntax

Browser messages are transported via the mailslot protocol, as specified in section 2.1. The browser message MUST be contained in the Data section of the mailslot transaction SMB, as specified in [MS-SMB] section 2.2.12.23.

Browser messages can be categorized according to the server's role. Each of these lists is complete for the specified, individual role. A machine that assumes multiple roles will use the messages for each of those roles, as described here:

- Messages used by non-browser servers are the following:
  - HostAnnouncement (sent).
  - AnnouncementRequest (received).
- Messages used by **browser clients** are the following:
  - GetBackupListRequest (sent).
  - GetBackupListResponse (received).
  - RequestElection (sent).
- Messages used by all browser servers (with subcategories of **potential browser server**, **backup browser servers**, local master browser servers, and domain master browser servers) are the following:

- RequestElection (sent, received).
- AnnouncementRequest (sent).

Messages used by potential browser servers are the following:

- All messages used by all browser servers.
- BecomeBackup (received).

Messages used by backup browser servers are the following:

- All messages used by all browser servers.
- LocalMasterAnnouncement (received).
- AnnouncementRequest (sent).

Messages used by Local Master Browser servers are the following:

- All messages used by all browser servers.
- HostAnnouncement (received).
- DomainAnnouncement (sent, received).
- AnnouncementRequest (sent).
- BecomeBackup (sent).
- GetBackupListRequest (received).
- GetBackupListResponse (sent).
- LocalMasterAnnouncement (sent).
- MasterAnnouncement (sent).

Messages used by Domain Master Browser servers are the following:

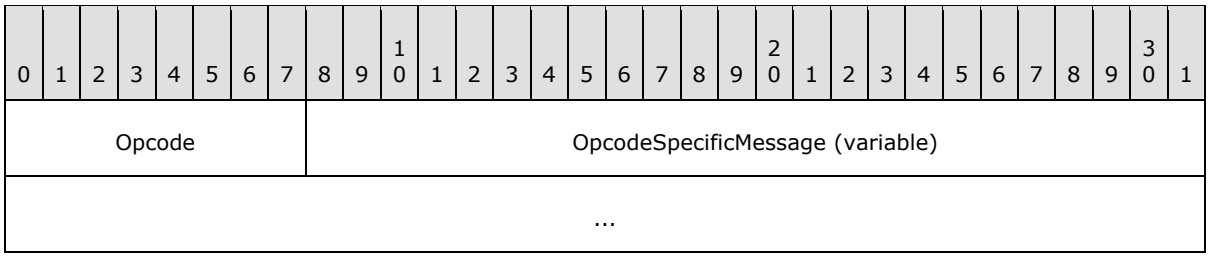
- All messages used by Local Master Browser servers.
- MasterAnnouncement (received).

More information about how the various browser server messages are used is specified in section 3.

All multibyte fields specified in messages in this section are transmitted in **little-endian** byte order, unless noted otherwise.

2.2.1 Browser Message Header

A browser message header consists of an 8-bit operation code (opcode) followed by data that depends on the opcode. The format of the header is as follows.



Opcode (1 byte): The opcode MUST be one of the values listed in the following table. Note: the sections that specify the messages are presented in opcode order.

Value	Meaning
HostAnnouncement 0x01	For more information, see section 2.2.2.
AnnouncementRequest 0x02	For more information, see section 2.2.3.
RequestElection 0x08	For more information, see section 2.2.4.
GetBackupListRequest 0x09	For more information, see section 2.2.5.
GetBackupListResponse 0x0A	For more information, see section 2.2.6.
BecomeBackup 0x0B	For more information, see section 2.2.7.
DomainAnnouncement 0x0C	For more information, see section 2.2.8.
MasterAnnouncement 0x0D	For more information, see section 2.2.9.
ResetStateRequest 0x0E	For more information, see section 2.2.10.
LocalMasterAnnouncement 0x0F	For more information, see section 2.2.11.

OpcodeSpecificMessage (variable): A variable-length browser message that MUST have the form specified in the section of this specification that is indicated in the preceding table.

2.2.2 HostAnnouncement Browser Frame

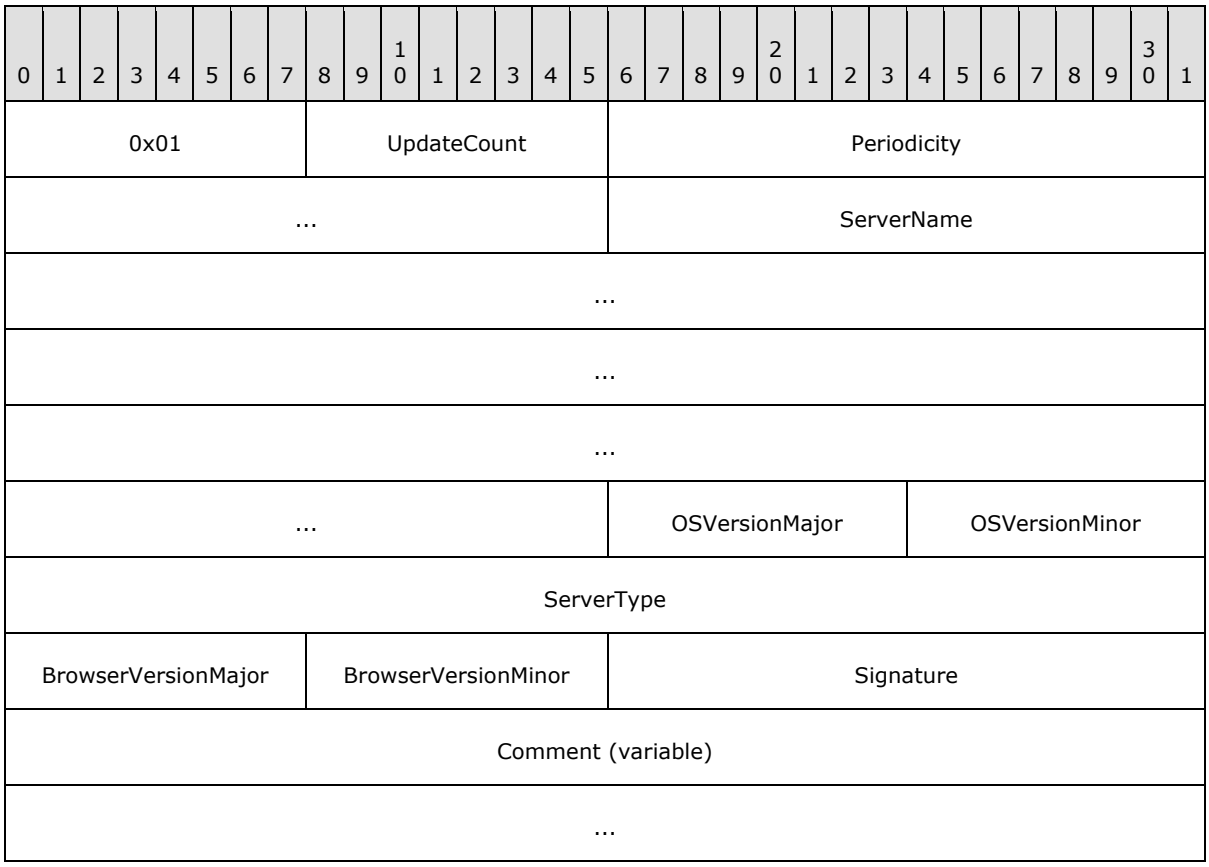
A server (including nonbrowser servers) sends a HostAnnouncement browser frame to advertise its presence and to specify the types of resources and services it supports. It MUST be a response to an AnnouncementRequest browser frame, as specified in section 2.2.3, or to the expiration of the HostAnnouncement timer, as specified in section 3.2.2.

A server MUST issue a HostAnnouncement in response to a received AnnouncementRequest browser frame (as defined in section 2.2.3) or as a response to the expiration of the announcement timer, as

specified in section 3.2.6. Failure to do so results in this server's resources being absent in the resource enumeration to browser clients.

This frame MUST be sent to mailslot \MAILSLOT\BROWSE or mailslot\MAILSLOT\LANMAN, or to the unique name <machine group>[1d] (that is, to the **local master browser**).

The format of the HostAnnouncement frame MUST be as follows.



0x01 (1 byte): The 8-bit operation code (opcode) that identifies this structure as a HostAnnouncement frame. This opcode MUST have a value of 0x01.

UpdateCount (1 byte): An unsigned 8-bit integer that MUST be sent as 0x00 and MUST be ignored on receipt.

Periodicity (4 bytes): An unsigned 32-bit integer that MUST be the announcement frequency of the server in milliseconds. It MUST be set to the NewHostAnnouncement timer value of the server in milliseconds, as specified in section 3.2.6.<6>

ServerName (16 bytes): MUST be a null-terminated ASCII server name with a length of 16 bytes including the null terminator. If the name is fewer than 16 bytes in length, including the terminator; the remainder of the 16 bytes MUST be filled with 0x00. The name <ServerName>[0x00] MUST be registered with NetBIOS by the server offering the services.

OSVersionMajor (1 byte): An unsigned 8-bit integer that MUST indicate the major version number of the operating system the server is running. This is entirely informational and does not have any significance for the browsing protocol.<7>

OSVersionMinor (1 byte): An unsigned 8-bit integer that MUST indicate the minor version number of the operating system the server is running. This is entirely informational and does not have any significance for the browsing protocol.<8>

ServerType (4 bytes): An unsigned 32-bit integer that MUST be the type of the server, as specified in [MS-RAP] section 2.5.5.2.1.

BrowserVersionMajor (1 byte): A signed 8-bit integer that MUST indicate the major version number of the CIFS Browser Protocol that the server is running. Announcing servers MUST set this to 0x0F. This is entirely informational and does not have any significance for the browsing protocol. This field MUST NOT be validated in any way on receive.

BrowserVersionMinor (1 byte): An unsigned 8-bit integer MUST indicate the minor version number of the CIFS Browser Protocol that the server is running. Announcing servers MUST set this to 0x01. This is entirely informational and does not have any significance for the browsing protocol. This field MUST NOT be validated in any way on receive.

Signature (2 bytes): An unsigned 16-bit integer that MUST be set to 0xAA55.

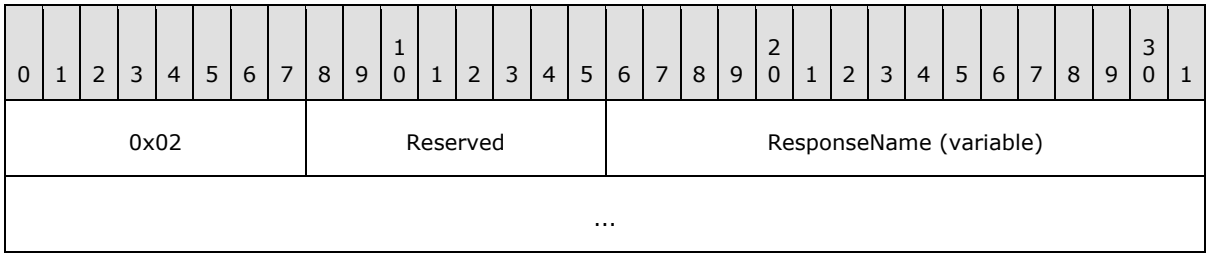
Comment (variable): A null-terminated ASCII string that MUST be less than or equal to 43 bytes in length including the null terminator. This is a purely informational comment associated with the server and has no effect on the operation of the CIFS Browser Protocol.<9>

2.2.3 AnnouncementRequest Browser Frame

The AnnouncementRequest frame MUST be sent from the NetBIOS computer name <computer>[0x00] to the NetBIOS group name <machine group>[0x00], to force all machines in the workgroup or domain to announce, or it MUST be sent from the NetBIOS computer name <computer>[0x00] to the NetBIOS group name <machine group>[0x1D], to force the current master browser on the subnet to announce itself to the client. It is sent by a local master browser server to <machine group>[0x00] at startup to discover the members of <machine group>, as specified in section 3.3.6. Its expected response is a set of HostAnnouncement frames, as specified in section 2.2.2.

The frame MUST be sent to mailslot \MAILSLOT\BROWSE.

The format of the AnnouncementRequest frame MUST be as follows.



0x02 (1 byte): The 8-bit operation code (opcode) that identifies this structure as an AnnouncementRequest frame. This opcode MUST have a value of 0x02.

Reserved (1 byte): This value MUST be 0x00.

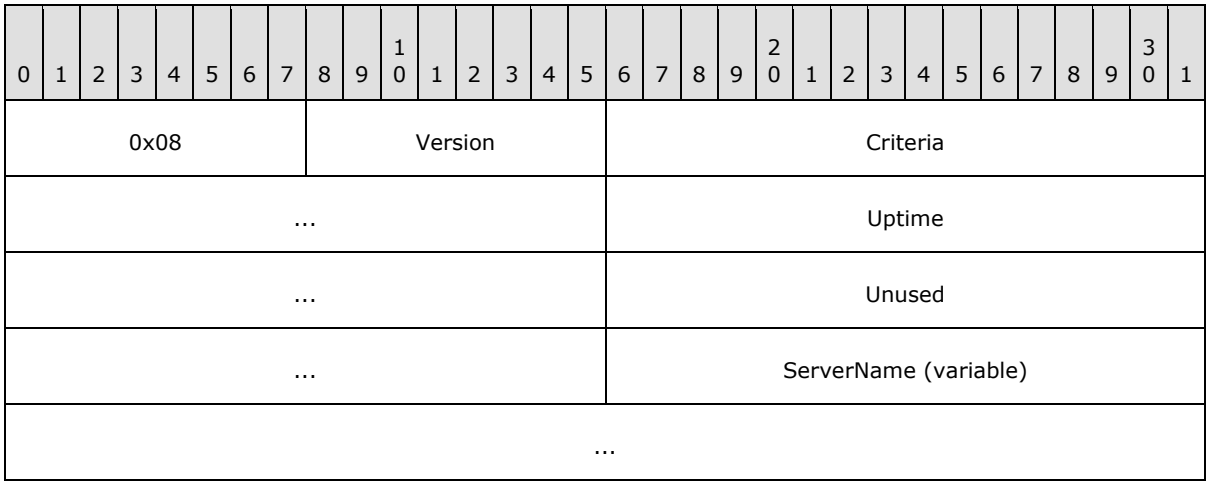
ResponseName (variable): A variable-length field that MUST be the name of the sender, up to 16 bytes in length including the null terminator. The receiving computer MUST ignore this

name. (Note that the name is not needed to generate a HostAnnouncement response because that message is sent as specified in section 2.2.2.)

2.2.4 RequestElection Browser Frame

The RequestElection frame MUST be broadcast by using the NetBIOS group name <machine group>[0x1E] and mailslot \MAILSLOT\BROWSE. For more information about browser elections, see sections 3.1.6, 4.2, 4.3, and 3.3.5.8.

The format of the RequestElection frame MUST be as follows.



0x08 (1 byte): The 8-bit operation code (opcode) that identifies this structure as a RequestElection frame. This opcode MUST have a value of 0x08.

Version (1 byte): An 8-bit integer that specifies the version of this election packet and MUST be transmitted with a value of 0x01.

Criteria (4 bytes): An unsigned 32-bit integer that MUST specify the **election criteria** of the sender. It MUST be produced by applying a bitwise OR operation on a combination of the appropriate operating system value, the browser version value, and one or more of the role values as listed in the following sections. The election process is specified in section 3.3.5.8.<10>

Operating System

Value	Meaning
0x00000000	Windows for Workgroups and Windows 95.
0x10000000	Windows Vista, Windows XP, Windows 2000 Professional, and Windows NT.
0x20000000	Windows Server 2008, Windows Server 2003, Windows 2000 Server, and Windows NT Server.

Browser Version

Value	Meaning
0x00000C00	Windows for Workgroups and Windows 95.
0x00010F00	Windows Vista, Windows XP, Windows 2000 Professional, and Windows NT. Windows Server 2008, Windows Server 2003, Windows 2000 Server, and Windows NT Server.

Role

Value	Meaning
0x00000080	Primary domain controller (PDC)
0x00000008	**Preferred master browser server**
0x00000004	A **master browser server** that is currently running.
0x00000002	A backup browser server that has been manually designated to be a backup browser server.<11>
0x00000001	A backup browser server that is currently running.
0x00000020	A computer using NetBIOS Name Service (or **Windows Internet Name Service**) for NetBIOS.

Uptime (4 bytes): An unsigned 32-bit integer that MUST be the number of seconds since the browser service was started on the server.

Unused (4 bytes): An unsigned 32-bit integer that MUST be sent as 0x00000000 and ignored on receipt.

ServerName (variable): MUST be a null-terminated ASCII server name and MUST be less than, or equal to, 16 bytes in length, including the null terminator.

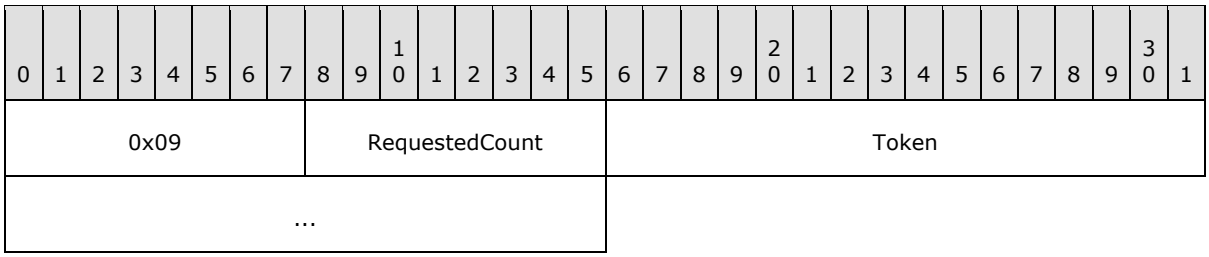
2.2.5 GetBackupListRequest Browser Frame

The GetBackupListRequest frame is sent by a browser client to the local master browser server, for a machine group to retrieve the identities of backup browser servers. Its response is a GetBackupListResponse frame, as specified in section 2.2.6. For more information about the use of the GetBackupListRequest frame, see section 3.1.5.1.

To get the list of backup browser servers for <machine group> from the local master browser server for that domain, the GetBackupListRequest browser frame MUST be sent to the master browser server with NetBIOS unique name <machine group>[0x1D] and mailslot \MAILSLOT\BROWSE.

To get the list of backup browser servers for <domain> from the domain master browser server for that domain, the GetBackupListRequest browser frame MUST be sent to the domain master browser server with the NetBIOS unique name <domain>[0x1B] and mailslot \MAILSLOT\BROWSE.

The format of the GetBackupListRequest frame, which is 6 bytes in length, MUST be as follows.



0x09 (1 byte): The 8-bit operation code (opcode) that identifies this structure as a GetBackupListRequest frame. This opcode MUST have a value of 0x09.

RequestedCount (1 byte): An 8-bit integer that specifies the number of backup servers that the client is requesting.<12>

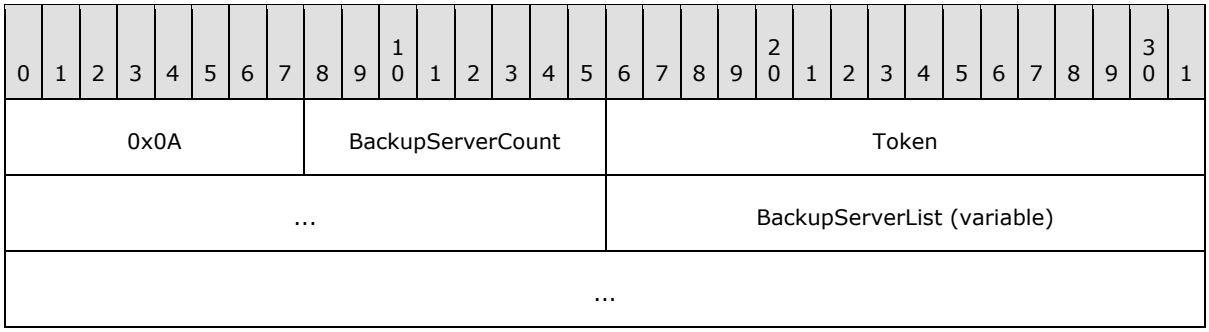
Token (4 bytes): MUST be a 32-bit value. This field has significance only to the client issuing the browser frame. The local master browser server MUST return this token unmodified in the corresponding GetBackupListResponse response message. The client MUST use this to distinguish replies to multiple outstanding GetBackupList requests.<13>

2.2.6 GetBackupListResponse Browser Frame

The GetBackupListResponse frame MUST be sent by a master browser server to the computer system that sends a GetBackupListRequest frame. It is a response to a GetBackupListRequest browser frame.

This frame MUST be sent to the NetBIOS unique name <computer>[0x00] and mailslot \MAILSLOT\BROWSE, where <computer> is the name of the originator of the GetBackupListRequest frame.

The format of the GetBackupListResponse frame MUST be as follows.



0x0A (1 byte): The 8-bit operation code (opcode) that identifies this structure as a GetBackupListResponse frame. This opcode MUST have a value of 0x0A.

BackupServerCount (1 byte): An unsigned 8-bit integer that MUST be the number of backup servers in the **BackupServerList** field.

Token (4 bytes): An unsigned 32-bit value that MUST be the token value received in GetBackupListRequest. The server MUST return the same value here.

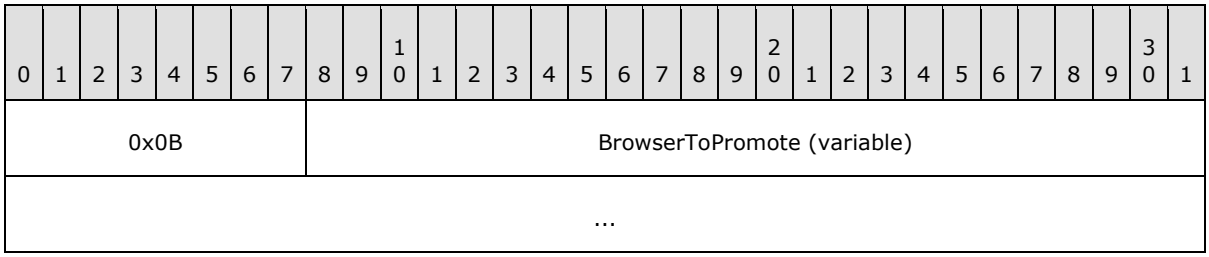
BackupServerList (variable): MUST be a series of null-terminated ASCII strings, each up to 16 bytes in length including the null terminator, where each string MUST denote a server

name acting as a backup browser server. The number of such strings present MUST be specified in BackupServerCount.

2.2.7 BecomeBackup Browser Frame

When a local master browser server for a machine group wants to promote a **potential browser server** to backup browser server, it MUST send a BecomeBackup frame by using the NetBIOS group name <machine group>[0x1E] and mailslot \MAILSLOT\BROWSE.

The definition of the BecomeBackup frame MUST be as follows.



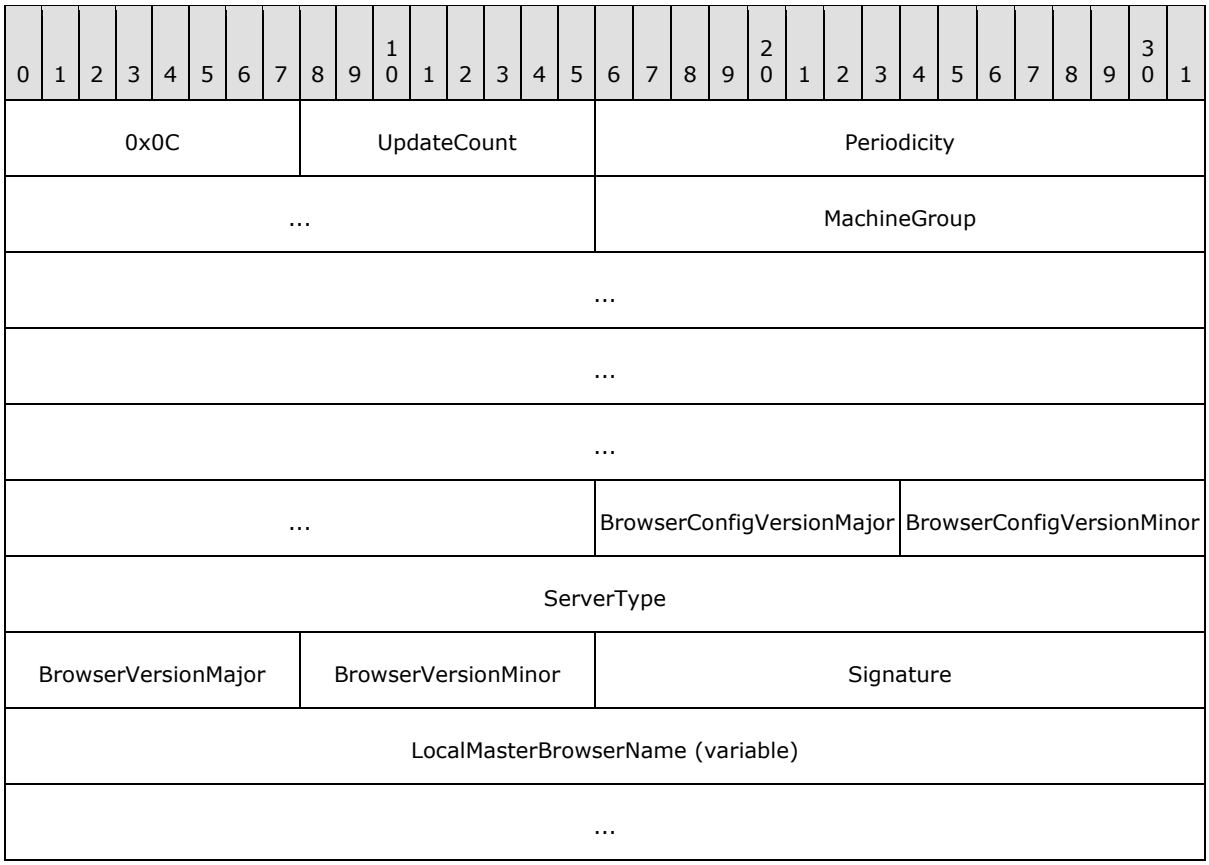
0x0B (1 byte): The 8-bit operation code (opcode) that identifies this structure as a BecomeBackup frame. This opcode MUST have a value of 0x0B.

BrowserToPromote (variable): MUST be a null-terminated ASCII string that is less than or equal to 16 bytes in length, including the null terminator, which MUST be the name of the browser server to be promoted to backup.

2.2.8 DomainAnnouncement Browser Frame

Local Master Browser servers announce the machine group they serve to any other Local Master Browser servers on their subnet by broadcasting a DomainAnnouncement frame using the NetBIOS group name [0x01][0x02]__MSBROWSE__[0x02][0x01] and mailslot \MAILSLOT\BROWSE.

The format of the DomainAnnouncement frame MUST be as listed in the following table.



0x0C (1 byte): The 8-bit operation code (opcode) that identifies this structure as a DomainAnnouncement frame. This opcode MUST have a value of 0x0C.

UpdateCount (1 byte): An unsigned 8-bit integer that MUST be sent as 0x00, and MUST be ignored on receipt.

Periodicity (4 bytes): An unsigned 32-bit integer that MUST be the announcement frequency, in milliseconds, of the machine group, as specified in section 3.3.2.

MachineGroup (16 bytes): MUST be a null-terminated ASCII workgroup or domain name with a length of 16 bytes, including the null terminator. If the name is less than 16 bytes in length, including the terminator, the remainder of the 16 bytes MUST be filled with 0x00.

BrowserConfigVersionMajor (1 byte): An unsigned 8-bit integer that SHOULD be set to the major version of the browser protocol that the server is running. This value is provided for informational purposes only and is irrelevant to the browsing protocol.<14>

BrowserConfigVersionMinor (1 byte): An unsigned 8-bit integer that SHOULD indicate the minor version of the browser protocol that the server is running. This value is provided for informational purposes only and is irrelevant to the browsing protocol.<15>

ServerType (4 bytes): An unsigned 32-bit integer that MUST be the type of the server. The server type bits MUST be set as specified in [MS-RAP] section 2.5.4.2.1.

BrowserVersionMajor (1 byte): An unsigned 8-bit integer that SHOULD have the value 0x0F.<16>

BrowserVersionMinor (1 byte): An unsigned 8-bit integer that SHOULD have the value 0x01.<17>

Signature (2 bytes): An unsigned 16-bit integer that MUST have the value 0xAA55.

LocalMasterBrowserName (variable): A null-terminated ASCII string that MUST contain the name of the sender, up to 16 bytes in length including the null terminator.

2.2.9 MasterAnnouncement Browser Frame

The MasterAnnouncement frame MUST be sent by a local master browser to the domain master browser when the MasterAnnouncement timer expires, as specified in section 3.3.6. The MasterAnnouncement frame MUST be sent to the NetBIOS unique name <PDCName>[0x00] and mailslot \MAILSLOT\BROWSE where <PDCName> is the computer name of the domain master browser.

The format of the MasterAnnouncement frame MUST be as follows.

0x0D (1 byte): The 8-bit operation code (opcode) that identifies this structure as a MasterAnnouncement frame. This opcode MUST have a value of 0x0D.

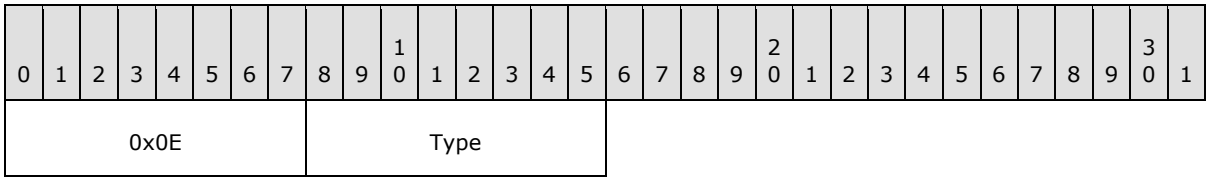
MasterBrowserServerName (variable): A null-terminated ASCII string that MUST contain the name of the local master browser and MUST be less than or equal to 16 bytes in length, including the null terminator.

2.2.10 ResetStateRequest Browser Frame

The ResetStateRequest frame instructs a browser server to change its operational state.

The local master browser SHOULD send a RESET_STATE_CLEAR_ALL request to any backup browser that is determined to be running an older version of the browser protocol when the HostAnnouncement Timer count-down reaches 0x00.

The format of the ResetStateRequest frame MUST be as listed in the following table.



0x0E (1 byte): The 8-bit operation code (opcode) that identifies this structure as a LocalMasterAnnouncement frame. This opcode MUST have a value of 0x0E.

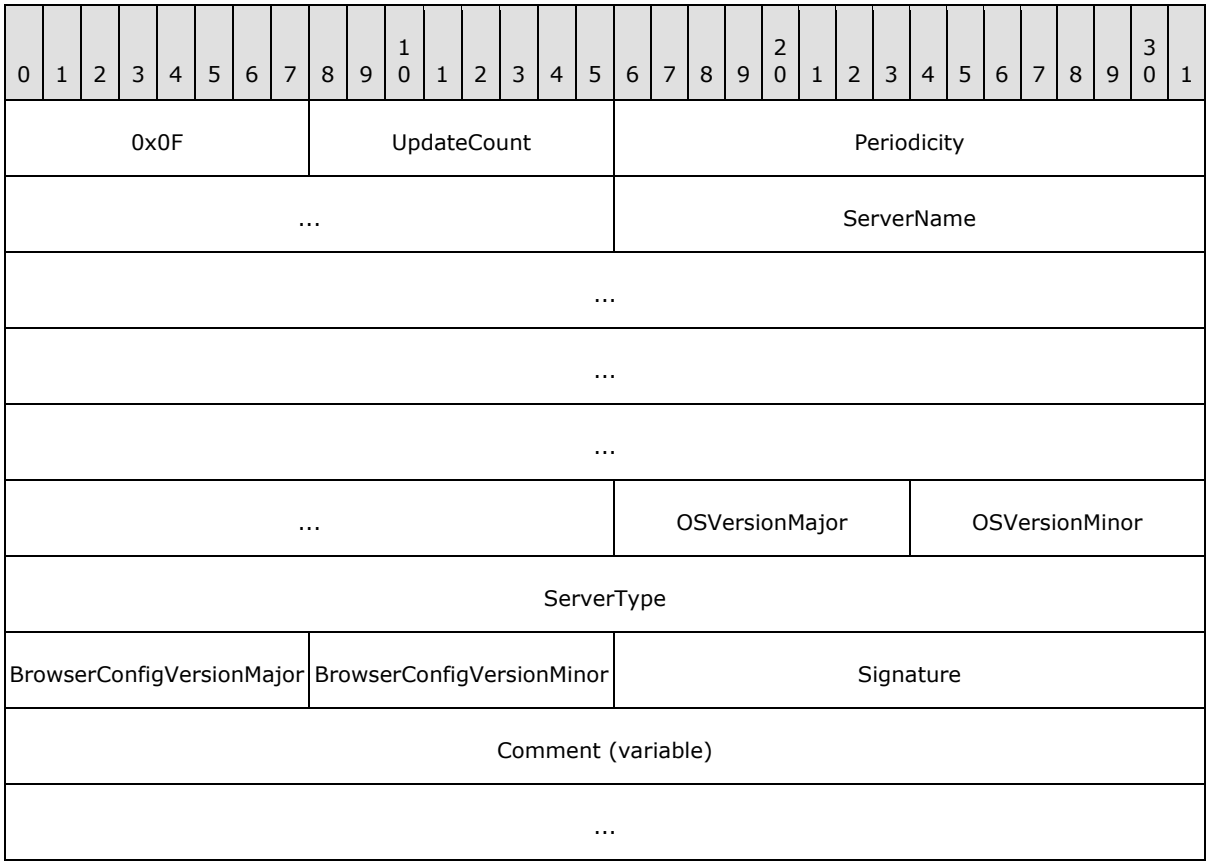
Type (1 byte): An unsigned 8-bit integer that MUST be set to one of the following, possible values:

Value	Meaning
RESET_STATE_STOP_MASTER 0x01	Instructs master browser server to change its role to browser server.
RESET_STATE_CLEAR_ALL 0x02	Instructs browser server to stop all browser server activities. It is disqualified from acting as a browser server.
RESET_STATE_STOP 0x04	Instructs browser server to stop the browser service.

2.2.11 LocalMasterAnnouncement Browser Frame

A local master browser for a machine group MUST announce itself with the periodicity listed in section 3.3.2 to all the other browser servers in its machine group that are on its subnet, using the LocalMasterAnnouncement frame. The LocalMasterAnnouncement frame MUST be broadcast by using the NetBIOS group name <machine group>[0x1E] and mailslot \MAILSLOT\BROWSE.

The format of the LocalMasterAnnouncement frame MUST be as listed in the following table.



0x0F (1 byte): The 8-bit operation code (opcode) that identifies this structure as a LocalMasterAnnouncement frame. This opcode MUST have a value of 0x0F.

UpdateCount (1 byte): An unsigned 8-bit integer that MUST be set to 0x00 and MUST be ignored upon receipt.

Periodicity (4 bytes): An unsigned 32-bit integer that MUST be the announcement frequency of the local master browser server in milliseconds. It MUST be set to the NewLocalMasterAnnouncement timer value of the server in milliseconds. For more information, see section 3.3.2.

ServerName (16 bytes): MUST be a null-terminated ASCII server name with a length of 16 bytes, including the null terminator. If the name is less than 16 bytes in length including the terminator, then the remainder of the 16 bytes MUST be filled with 0x00.

OSVersionMajor (1 byte): MUST be an unsigned 8-bit integer that indicates the major version of the operating system that the server is running. This value is provided for informational purposes only and is irrelevant to the browsing protocol.<18>

OSVersionMinor (1 byte): MUST be an unsigned 8-bit integer that indicates the minor version of the operating system the server is running. This value is provided for informational purposes only and is irrelevant to the browsing protocol.<19>

ServerType (4 bytes): An unsigned 32-bit integer that MUST be the type of the local master browser server. The type bits are as specified in [MS-RAP] section 2.5.5.2.1.

BrowserConfigVersionMajor (1 byte): An unsigned 8-bit integer that SHOULD be set to the major version of the browser protocol that the server is running. This value is legacy and is irrelevant to the browsing protocol.<20>

BrowserConfigVersionMinor (1 byte): An unsigned 8-bit integer that SHOULD be set to the major version of the browser protocol that the server is running. This value is legacy and is irrelevant to the browsing protocol.<21>

Signature (2 bytes): An unsigned 16-bit integer that MUST have the value 0xAA55.

Comment (variable): A null-terminated ASCII string that MUST be less than or equal to 43 bytes in length, including the null terminator. This is a purely informational comment associated with the server and has no effect on the operation of the Browser Protocol.

3 Protocol Details

The hosts that are used in the browsing process can be separated into four distinct groups:

- Browser clients
- Nonbrowser servers
- Browser servers
- Domain master browser servers

3.1 Client Details

3.1.1 Abstract Data Model

This section describes a hypothetical model of browser client data organization that can be implemented to support the CIFS Browser Protocol. The purpose of this description is to help explain how this aspect of the protocol works. This specification does not prescribe that implementations adhere to this model as long as their external behavior is consistent with what is described throughout this document.

One objective of this model is to minimize the cost of locating backup browser servers. Every time a browser client successfully receives a response to a **NetServerEnum2** request, as specified in [MS-RAP] section 2.5.5.2, a backup browser server MUST be updated with the contents of the response.

For this purpose, a browser client maintains the following state.

BrowserServerList: The BrowserServerList is a cache of browser servers in the machine group of the client.

3.1.2 Timers

The client uses the following timer:

GetBackupListRequest Timer: This timer is used to govern the retransmission of GetBackupListRequest frames. Its initial duration MUST be 1 second.

3.1.3 Initialization

At startup, to find a backup browser server, the client MUST do the following:

1. Initialize GetBackupListRequest timer and send a GetBackupListRequest frame, as specified in section 2.2.5.
2. Obtain a list of backup browser servers.
3. Cache an implementation-defined number of servers chosen from this list.<22>

In case the local master browser for a machine group fails to respond to the GetBackupListRequest after an implementation-defined number of retries, as specified in section 3.1.6, the client MUST force an election by sending a RequestElection frame.

The GetBackupListRequest and GetBackupListResponse sequences are specified in sections 3.1.5.1.1 and 3.1.5.1.2, respectively. If this sequence does not produce a backup browser server, as specified in section 3.1.6, the initialization MUST fail.<23>

3.1.4 Higher-Layer Triggered Events

When a higher layer (such as the client Presentation or Application layer) issues a request for the set of servers for a specified machine group, the client MUST behave as specified in this section.

First, if the client has no backup browser servers in its list for the machine group, or no list at all, it MUST obtain a list, as specified in section 3.1.5.1.

If the client is able to populate a list of backup browser servers, it MUST then select a backup server at random from its list for that machine group, and send it a **NetServerEnum2** request, as specified in [MS-RAP] section 2.5.5.2. The objective is to enable multiple backup browser servers to effectively handle high browsing loads. If the client is not able to populate the server list, or an error occurs while retrieving the server list, it MUST return the error to the higher layer.

3.1.5 Message Processing Events and Sequencing Rules

A browser client MUST ignore all CIFS Browser Protocol messages except GetBackupListRequest.

3.1.5.1 Retrieving a List of Backup Browser Servers

When a browser client needs to determine the set of backup browser servers for a particular machine group, the browser client MUST send a GetBackupListRequest frame and check whether it receives a GetBackupListResponse frame.

3.1.5.1.1 Sending a GetBackupListRequest Frame

A browser client sends a GetBackupListRequest frame, as specified in section 2.2.5. When generating the GetBackupListRequest, the client MUST generate a token that is unique within the client. How the client selects the token is implementation-defined. The token exists solely to allow the client to differentiate between GetBackupListResponse calls.<24>

The browser client MUST send the frame to <machine group>[0x1D], where <machine group> is the domain or workgroup within which the client is operating.

After the browser request has been sent, the client MUST start the GetBackupListRequest timer.

3.1.5.1.2 Receiving a GetBackupListResponse Frame

After the local master browser server responds with a list of backup browser servers, the client SHOULD choose an implementation-defined number of servers from within the response by using an implementation-dependent algorithm, and then cache them.<25>

When a GetBackupListResponse frame is received, the corresponding timer MUST be stopped. Because a client can be a member of only a single machine group, it needs only one timer.

3.1.6 Timer Events

When a GetBackupListRequest timer expires without receiving a GetBackupListResponse, the GetBackupListRequest frame MAY be retransmitted. The delay MUST be at least twice the expected service time, which MUST be 1 second.<26>

In case the local master browser for a machine group fails to respond to the GetBackupListRequest after an implementation-defined number of retries, the client MUST force an election by sending a RequestElection frame (for more information, see section 2.2.4). If the client is unable to retrieve a list of browser servers from the local master browser server, it MAY attempt to retrieve a list of backup browser servers by sending a GetBackupListRequest frame directly to the domain master

browser for that domain by using the unique name <domain>[0x1B] that is registered by the domain master browser. The default value of the retry count MUST be 3.<27>

3.1.7 Other Local Events

None.

3.2 Nonbrowser Server Details

3.2.1 Abstract Data Model

This section describes a hypothetical model of nonbrowser server data organization that can be implemented to support the CIFS Browser Protocol. The purpose of this description is to help explain how this aspect of the protocol works. This specification does not prescribe that implementations adhere to this model, as long as their external behavior is consistent with the behavior described throughout this document.

A nonbrowser server MUST implement the abstract data model for a server, as specified in [MS-RAP] section 3.2.1.4.

In addition to that information, the nonbrowser server MUST implement the following:

Server.HostAnnouncementCount: The number of times the HostAnnouncement timer (as specified in 3.2.2) has expired.

3.2.2 Timers

Nonbrowser servers use the following conceptual timers.

HostAnnouncement timer: Used to periodically advertise itself to the local master browser for its machine group. For more information about the HostAnnouncement timer, see section 3.2.6.

AnnouncementRequest response timer: Used to delay responding to an AnnouncementRequest. For more information, see section 3.2.5.1.

3.2.3 Initialization

When a nonbrowser server starts up, it MUST start the HostAnnouncementTimer, as specified in section 3.2.2. Whenever the HostAnnouncementTimer fires, the nonbrowser server MUST issue a HostAnnouncement frame.

A nonbrowser server MUST register the NetBIOS name <machine group>[0x1D] corresponding to the domain or workgroup within which the nonbrowser server resides. This makes it possible to receive AnnouncementRequest browser frames.

All other browser messages sent to the nonbrowser server with different NetBIOS names MUST be ignored.

The name <ServerName>[0x00] MUST be registered with NetBIOS by the server offering the service.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

A nonbrowser server MUST ignore all CIFS Browser Protocol messages except the AnnouncementRequest browser frame.

3.2.5.1 Receiving an AnnouncementRequest Frame

On receiving an AnnouncementRequest frame, a nonbrowser server MUST generate a random number in the range [0, 30] seconds. It MUST then set its AnnouncementRequest response timer to that value.

3.2.6 Timer Events

When either the HostAnnouncement or AnnouncementRequest timer expires, a nonbrowser server MUST send a HostAnnouncement frame that specifies the type of resources or services that it is advertising. When the HostAnnouncement timer fires, it SHOULD reset the HostAnnouncement timer based on the following table.<28>

Server.HostAnnouncementCount value (as specified in section 3.2.1)	New HostAnnouncement timer value
1	1 minute
2	2 minutes
3	4 minutes
4	8 minutes
> 4	12 minutes

3.2.7 Other Local Events

A nonbrowser server MUST send a HostAnnouncement frame that specifies a server type of zero, just prior to shutting down to allow it to be quickly removed from the list of available servers.

3.3 Browser Server Details

A browser server MUST follow all the rules for a nonbrowser server, in addition to the rules specified in this section. A browser server MUST follow the state machine below.

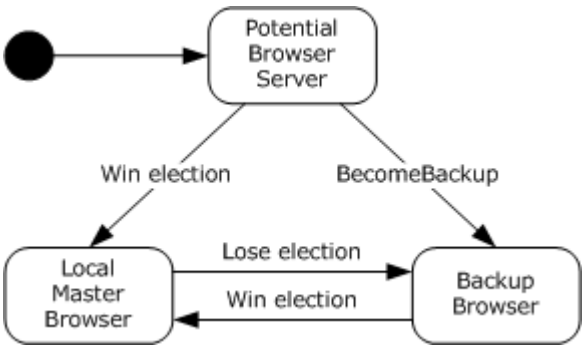


Figure 1: Browser server state machine

For all states, the SV_TYPE_POTENTIAL_BROWSER (PB) flag MUST be set in the **ServerType** field in a browser frame such as a HostAnnouncement frame. In the local master browser and backup browser states, an additional flag MUST be set as shown in the following table.

State	ServerType flag set
Backup browser	SV_TYPE_BACKUP_BROWSER (BB)
Local master browser	SV_TYPE_MASTER_BROWSER (MB)

When a local master browser server starts up, its server list may be empty; therefore, it MAY force all browser servers to announce themselves. The local master browser server does this by broadcasting an AnnouncementRequest (section 2.2.3) browser frame.<29>

The AnnouncementRequest frame MUST be broadcast by using the NetBIOS group name <machine group>[0x00]. The frame MUST be sent to the mailslot \MAILSLOT\BROWSE.

For more details regarding ServerType flag values, see [MS-SRVS] section 2.2.2.7.

3.3.1 Abstract Data Model

This section describes a sample model of browser server data organization that can be implemented to support this protocol. The purpose of this description is simply to help explain how this aspect of the protocol works. This specification does not prescribe that implementations adhere to this model as long as their external behavior is consistent with what is described throughout this document.

Backup Browser List: A list of the machines that the Local Master browser has designated as Backup Browser servers.

Current Role: The current state in the browser server state machine shown above.

Machine Groups List: An ordered list of machine groups; each entry MUST contain a domain or workgroup name, the name of the Local Master browser for that machine group (as given in the DomainAnnouncement frame specified in section 2.2.8) and a conceptual expiration timer for each machine group. The ordering of the list is implementation defined, but the order MUST be stable, and the list MUST NOT contain entries with duplicate names. The Machine Groups list MUST be ordered to implement the semantics of the NetServerEnum3 request (which specifies a resume point). For details of NetServerEnum2 and NetServerEnum3 as they are used here, see section 3.3.5.6.<30>

Election Transmission Count: A count of the number of RequestElection frames sent by the browser server during an election. The Election Transmission count MUST be reset on each election.

Domain Announcement Count: A count of the number of times the DomainAnnouncement timer (for more information, see section 3.3.2) has fired.

Preferred Flag: A configuration option indicating if the machine is to be preferred in browser server elections. Preferred **browsers** MUST operate as Backup Browser servers.

Servers List: An ordered list of servers and their associated information (as specified in section 3.2.1), as provided in HostAnnouncement frames. The ordering of the list is implementation defined, but the order MUST be stable, and the list MUST NOT contain entries with duplicate names. The Servers list must be ordered to implement the semantics of the NetServerEnum3 request (which specifies a resume point).<31>

Be aware that the above model can be implemented using a variety of techniques. An implementation can implement such data in any way.

3.3.2 Timers

DomainAnnouncement Timer: Used by a local master browser to periodically announce itself to local master browser servers of other machine groups on the subnet by sending a DomainAnnouncement frame, as specified in section 2.2.8. For more information about the DomainAnnouncement timer, see section 3.3.6.<32>

DomainAnnouncement Timer Count: A count of the number of times that the DomainAnnouncement timer has expired. It MUST be reset to 0 after a browser wins an election.

Machine Group Expiration Timer: For each entry in the Machine Groups list that is created or updated via DomainAnnouncements, the browser server keeps a conceptual expiration timer. This timer MUST be initialized to the **Periodicity** field value found in the DomainAnnouncement.

Election Delay Timer: The browser server keeps an Election Delay timer for use in elections. The value is specified in section 3.3.5.8.

FindMaster Timer: Used when a browser server must find a local master browser for the machine group of the browser server. This timer value MUST be 1500 milliseconds (msec).

LocalMasterAnnouncement Timer: Used to periodically advertise the local master browser to all the machines in the machine group on the local subnet by sending a LocalMasterAnnouncement (section 2.2.10) frame. For more information about the LocalMasterAnnouncement timer, see section 3.3.6.<33>

LocalMasterAnnouncement Timer Count: A count of the number of times that the LocalMasterAnnouncement timer has expired. It MUST be reset to 0 after a browser wins an election.

MasterAnnouncement Timer: Used to periodically advertise the local master browser to the domain master browser by sending a MasterAnnouncementBrowser (section 2.2.9) frame. For more information about the MasterAnnouncementBrowser timer, see section 3.3.6 . If the local master browser is not a member of a domain, this timer MUST be ignored. The default value for this timer MUST be 5 minutes.<34>

MasterAnnouncement Timer Count: A count of the number of times that the MasterAnnouncement timer has expired. It MUST be reset to 0 after a browser wins an election.

NetServerEnum2 Timer: Used to periodically allow the backup browser server to refresh its list of servers from the local master browser server, or local master browser servers to refresh their server list from the domain master browser server. The NetServerEnum2 timer SHOULD control the accuracy of the information in the browser server lists. If the NetServerEnum2 timer duration is low, the information in the servers list SHOULD be more accurate, but the load on the local master browser server (or domain master browser server) MAY be higher. If the NetServerEnum2 timer duration is high, the information in the Servers list SHOULD be less accurate, but the load on the local master browser server (or Domain Master Browser server) MAY be higher. The default value for this timer is 5 minutes.<35>

Server Expiration Timer: For each server entry in the Servers list that is created or updated via HostAnnouncements, the browser server keeps a conceptual expiration timer. This timer MUST be initialized to the **Periodicity** field value found in the HostAnnouncement.

3.3.3 Initialization

If the Preferred flag is set, the machine MUST force an election by sending a RequestElection (section 2.2.4) frame. It MUST set the Preferred Master bit in the Criteria in all RequestElection frames it sends.

If the Preferred flag is not set, the machine MUST start up as a potential browser server. It MUST announce itself by sending a HostAnnouncement frame, and if it receives a BecomeBackup frame, it MUST become a backup server.

The browser server MUST register NetBIOS names <machine group>[0x00] and <machine group>[0x1E]. Information on how to register NetBIOS names is as specified in [RFC1001].

3.3.4 Higher-Layer Triggered Events

If a browser server gets promoted to primary domain controller, it MUST force an election by sending a RequestElection frame, and assume the duties of a domain master browser, as specified in section 3.4.

3.3.5 Message Processing Events and Sequencing Rules

After receiving a CIFS Browser Protocol frame, the operation code (opcode) MUST first be inspected to determine the message type. If the opcode is not defined in this specification, the frame MUST be silently ignored. If the opcode is recognized, the browser server MUST then determine if the message is correctly formatted as specified in section 2.2. Because messages are transmitted as datagrams, malformed messages MUST be silently ignored.<36>

A browser server MUST ignore the GetBackupListResponse frame. Correctly formed frames MUST then be processed as specified in the following subsections.

3.3.5.1 Receiving a BecomeBackup Frame

The local master browser server MUST send a BecomeBackup frame to a potential browser server when it determines that the number of current backup browser servers for the machine group on the local master browser server's subnet does not meet the criteria as specified in section 3.3.5.7.

A browser server that receives a BecomeBackup frame MUST attempt to become a backup browser if it is still willing to be.

Note If the browser server is not willing to become a backup, for example, because it is overloaded, the browser server MUST send a new HostAnnouncement (section 2.2.2), indicating that it is no longer a potential browser.

If the browser server does not know the name of the local master browser for its machine group, it MUST send an AnnouncementRequest frame request to <machine group>[0x1D], and start the FindMaster timer. If the FindMaster timer expires before the server receives a LocalMasterAnnouncement frame, the browser server MUST issue another AnnouncementRequest frame request to <machine group>[0x1D] and reset the FindMaster timer. If the server issues an implementation-defined number of FindMaster frame requests without receiving a LocalMasterAnnouncement frame response, then the server MUST send a RequestElection (section 2.2.4) frame to force the election of a new master browser. For more information on the election process, see section 3.3.5.8.<37>

If the browser server does not have a server list, it MUST attempt to retrieve a list of servers from the master browser for the machine group. If successful, the new backup browser MUST immediately send a HostAnnouncement frame with the **Flags** field in the announcement reflecting the new state of the browser.

Note that after the master browser receives the HostAnnouncement frame, it MUST hand out this browser server name in GetBackupListResponse frames, and clients will contact this browser server as if it is a backup browser server.

A nonbrowser server that receives a BecomeBackup frame MUST ignore the frame, as specified in section 3.2.5. Similarly, a backup browser server that receives a BecomeBackup frame MUST ignore the frame.

3.3.5.2 Receiving a LocalMasterAnnouncement Frame

A browser server can discover the master browser for a machine group by issuing an AnnouncementRequest (section 2.2.3) frame to the name <machine group>[0x1D].

A LocalMasterAnnouncement frame MUST be processed as follows:

- If the browser server is in the BackupBrowser state, it MUST update the name of the local master browser that sent it in the Machine Groups list, adding a new entry if one does not exist.
- If the browser server is in the local master browser state, and the sender claims to be the local master browser (meaning the SV_TYPE_MASTER_BROWSER (MB) flag is set) for the same machine group, the browser server MUST demote itself from local master browser to become a Backup browser and force an election by sending a RequestElection frame. See section 3.3.5.8 for details about the election process.

3.3.5.3 Receiving a HostAnnouncement Frame

Nonbrowser servers and browser servers periodically (see section 3.2.6) send HostAnnouncement frames to inform the local master browser for the machine group about the status of the server.

Browser servers receiving the HostAnnouncement frame that are not the local master browser MUST ignore the frame.

A local master browser that receives a HostAnnouncement frame with the SV_TYPE_MASTER_BROWSER (MB) flag set MUST demote itself from local master browser to backup browser server and MUST force an election by sending a RequestElection (section 2.2.4) frame. For more information on the election process, see section 3.3.5.8.

The local master browser that receives this request MUST update the Servers list with the information included in the HostAnnouncement. The local master browser MUST reset the conceptual Server Expiration timer.

Upon receipt of a HostAnnouncement, if the number of backup browser servers is not sufficient for the machine group, the local master browser SHOULD send out BecomeBackup (section 2.2.7) packets according to the behavior as specified in section 3.3.5.7.

3.3.5.4 Receiving a DomainAnnouncement Frame

Local Master Browser servers periodically send HostAnnouncement frames to inform the local master browsers for other machine groups on the subnet about the status of the machine group. For more details on the timers related to this process, see section 3.2.6).

Browser servers receiving the DomainAnnouncement frame that are not the local master browser MUST ignore the frame.

The local master browsers receiving the request MUST update the machine groups list with the information included in the DomainAnnouncement. The local master browser MUST reset the conceptual machine group Expiration timer.

3.3.5.5 Receiving a GetBackupListRequest Frame

A browser server that is not the local master browser MUST ignore this request.

The local master browser MUST reply with a GetBackupListResponse frame that contains a list of the browser servers for its machine group, as specified in section 3.3.5.9.

3.3.5.6 Receiving a NetServerEnum2 or NetServerEnum3 Request

Browser clients issue NetServerEnum2 or NetServerEnum3 Remote Administration Protocol requests, as specified in [MS-RAP], to retrieve the list of servers or machine groups.

Browser servers receiving the NetServerEnum2 or NetServerEnum3 request that are not backup browser servers or local master browser servers MUST respond with Win32ErrorCode set to ERROR_REQ_NOT_ACCEP (as specified in [MS-GLOS]).

If the incoming NetServerEnum2 or NetServerEnum3 request specifies the LL flag in the *ServerType* parameter (as specified in [MS-RAP] section 2.5.5.2.1, the browser server MUST restrict the list of servers and machine groups returned to the client to the servers and machine groups on the same subnet as the browser server.

If the *ServerType* parameter of the incoming NetServerEnum2 or NetServerEnum3 request specifies the DL flag, and any other field in the **ServerType** field is set, the browser server MUST respond with the Win32ErrorCode set to ERROR_INVALID_FUNCTION (as specified in [MS-GLOS]).

If the incoming NetServerEnum2 or NetServerEnum3 request specifies the DL flag in the *ServerType* parameter (as specified in [MS-RAP] section 2.5.5.2.1, the browser server MUST return the server's list of machine groups to the client.

If the incoming NetServerEnum2 or NetServerEnum3 request specifies the *ServerType* as 0xFFFFFFFF, the browser server MUST return the complete list of servers to the client.

If the incoming NetServerEnum2 or NetServerEnum3 request specifies the *ServerType* as any value other than the above, the server MUST only return those servers with a **ServerType** field that contains one of the values in the request *ServerType* parameter.

If the browser server is processing a NetServerEnum3 request, it MUST return entries starting from its ordered list of servers or machine groups beginning with the server whose name matches the *FirstServerToReturn* parameter of the RAP NetServerEnum3Request packet, as specified in [MS-RAP].

If the request is for a list of servers in a domain that is different from the machine group that it serves, the local master browser MAY issue a NetServerEnum2 (or NetServerEnum3) request to the domain master browser for the specified domain (which it can find in its list of machine groups and their domain master browser servers). The domain master browser returns a NetServerEnum response that MUST then be returned to the requester.<38>

3.3.5.7 Sending BecomeBackup Frames

A local master browser server MUST choose the number of browser servers that will operate in the BackupBrowser server role. The number of backup browser servers is a trade-off between:

- Minimizing network traffic.
- Ensuring robustness by having multiple backup browser servers.
- Ensuring that when a local master browser server fails, there are multiple backup browser servers, which can become local master browser servers.<39>

If the local master browser server determines that one or more backup browser servers should be added to its server list, it MUST send BecomeBackup (section 2.2.7) frames to enough servers to get up to the recommended level of backup servers. Each server to which it sends a BecomeBackup frame MUST be in its server list and MUST NOT currently be a backup browser. (These new servers MUST be added to the backup browser list after receiving a HostAnnouncement from the browser server, indicating that its role has changed.)

3.3.5.8 Receiving a RequestElection Frame

The RequestElection frame (as specified in section 2.2.4) MUST be sent whenever a browser client or server is unable to retrieve information that is maintained by the local master browser server. It also MUST be issued when a local master browser server receives a frame that indicates that another machine on the subnet also believes it is a local master browser server.

When a browser server receives a RequestElection frame, it MUST calculate its election criteria and Uptime values, as specified in section 2.2.4.

The browser server MUST then compare its election criteria value with the election criteria value of the RequestElection frame as an unsigned 32-bit integer. If the browser server's election criteria is greater than the RequestElection frame, the browser server has "won" the election. If the browser server's election criteria is less than the RequestElection frame, the browser server has "lost" the election.

If the browser server's election criteria value is equal to the election criteria of the request frame, then the browser server MUST compare its **Uptime** field with the **Uptime** field of the RequestElection frame. If the browser server's Uptime value is greater than the **Uptime** value of the RequestElection frame, the browser server has "won" the election. If the browser server's **Uptime** value is less than the **Uptime** value of the RequestElection frame, the browser server has "lost" the election.

If the browser server's election criteria and **Uptime** are equal to the election criteria and **Uptime** of the Election Request, the browser server MUST compare its name with the name in the RequestElection frame. If the browser server's name is alphabetically less than the name in the RequestElection frame, the browser server has "won" the election. If the browser server's name is alphabetically greater than the name in the RequestElection frame, the browser server has "lost" the election.

If the browser server has "won" the election, the browser server MUST set its Election Transmission count to 0, and MUST set its Election Delay timer as shown in the following table.

Browser role	Election delay timer
Local master browser	100 ms

Browser role	Election delay timer
Backup browser	A pseudo-random number chosen from the range 200 ms to 600 ms
Potential browser	A pseudo-random number chosen from the range 800 ms to 3000 ms

In a domain environment the master browser server SHOULD query the primary domain controller that is acting as the domain master browser server for a list of other domains on which to listen for announcements as specified in section **3.1.5.1** of [MS-BRWSA]. If the list is not empty, the server SHOULD register the **NetBIOS names** <other domain name>[0x00] and accept requests on that name as described in section 3.3.3.

This election algorithm continues to execute as specified in section 3.3.6.

If the browser server has "lost" the election, the browser MUST stop its Election Delay timer, and if it was previously the local master browser, it MUST do the following:

- It MUST unregister the NetBIOS unique name <machine group>[0x1D] so that the winning browser server can successfully register it.
- It MUST take on the backup browser role.

3.3.5.9 Sending a GetBackupListResponse Frame

The **BackupServerCount** field in the GetBackupListResponse frame MUST be set to the number of entries in the backup browser list. The BackupServerList in the frame MUST be filled with the sequence of null-terminated server names from the BackupServerList.

The frame MUST be sent to the NetBIOS name <computer>[0x00] where <computer> is the name of the machine that sent the GetBackupListRequest.

3.3.6 Timer Events

DomainAnnouncement timer: When the DomainAnnouncement timer expires and the machine is a local master browser server, it MUST send a DomainAnnouncement (section 2.2.8) and it MUST reset the timer according to the following table.

DomainAnnouncement timer count value	New DomainAnnouncement timer value
1	1 minute
2	1 minute
3	5 minutes
4	5 minutes
5	10 minutes
6	10 minutes
> 6	15 minutes

Election Delay Timer: When this timer expires, the browser server MUST send a RequestElection frame, as specified in section 2.2.4, and increment the Election Transmission count.

If this counter is less than 4, and the browser server has "won" the election, the browser server MUST reset the Election Delay timer as shown in the following table.

Browser role	Election delay timer
Local master browser	100 msec
Backup browser/Potential browser	A pseudo-random number chosen from the range 200 msec to 600 msec
Potential browser	A pseudo-random number chosen from the range 800 msec to 3000 msec

If the Election Transmission count is greater than 30, the browser server MUST consider the election as lost. If the browser server is a local master browser server, the browser server MUST take the actions for losing an election, as specified in section 3.3.5.8.

When this counter reaches 4, the browser server MUST consider itself to have won the election, and MUST perform the following actions:

- The browser server MUST locate the domain master browser server. For more information, see [MS-ADTS].
- The browser server MUST send a MasterAnnouncement frame (as specified in section 2.2.9) to the domain master browser server and start the MasterAnnouncement timer.
- A newly elected local master browser server that has an empty Servers list MUST send an AnnouncementRequest frame, as specified in section 2.2.3.
- A local master browser server MUST register the NetBIOS unique name <machine group>[0x1D]. If the NetBIOS name registration fails, the browser server MUST initiate a new election by sending a new ElectionRequest frame. For more information, see section 2.2.4.
- If the Servers list is not empty, the local master browser server MUST initialize its backup browser server list by enumerating the servers in the server list for servers with the BB flag set (as specified in [MS-RAP] section 2.5.5.2.1. If there are not sufficient backup browsers to meet the criteria, as specified in section 3.3.5.7, the master browser server MUST send BecomeBackup frames to the potential browsers, as specified in section 3.3.5.7.

LocalMasterAnnouncement Timer: When the LocalMasterAnnouncement timer expires, and the machine is a local master browser server, it MUST announce itself to all browser servers for its machine group on its subnet by sending a LocalMasterAnnouncement frame, as specified in section 2.2.10, and it SHOULD reset the timer according to the following table.<40>

LocalMasterAnnouncement timer count Value	New LocalMasterAnnouncement timer value
1	1 minute
2	2 minutes
3	4 minutes
4	8 minutes

LocalMasterAnnouncement timer count Value	New LocalMasterAnnouncement timer value
> 4	12 minutes

MasterAnnouncement Timer: When the MasterAnnouncement timer expires, and the machine is a local master browser server, it MUST send a MasterAnnouncement, as specified in section 2.2.9, to its domain master browser server, and it MUST reset the MasterAnnouncement timer.

NetServerEnum2 Timer: When the NetServerEnum2 timer expires, and the machine is a backup browser server, the machine MUST send a NetServerEnum2 request to the local master browser server on its subnet for its machine group, to get a list of servers in that machine group, as well as a list of machine groups. It MUST then reset the NetServerEnum2 timer.

If the NetServerEnum2 request fails twice in succession, the backup browser server MUST send a RequestElection frame (as specified in section 2.2.4). For more information on the election process, see section 3.3.5.8.

If the machine is instead a local master browser server, and is a member of the domain, it MUST ask the domain master browser server for a domain-wide list of servers by issuing a NetServerEnum2 request with a ServerType parameter of 0xFFFFFFFF to retrieve the list of servers (as specified in section 3.3.5.6). This request retrieves the complete list of servers within the domain. The local master browser server then MUST issue the same request with the DL flag specified to retrieve the list of the machine groups. It MUST merge the results with its own list of servers and machine groups. The criteria for merging results are implementation dependent.<41>

Server Expiration Timer: When the expiration timer for a server in the servers list expires, the server MUST be removed from the servers list, if the local master browser has not received a HostAnnouncement request from that server for more than three times the periodicity specified by the server in the most-recently received HostAnnouncement frame. The server MUST NOT be removed from the Servers list before the Periodicity field in the last HostAnnouncement frame received from the server has elapsed.

If the server being removed from the Servers list is a member of the backup browser server list, the local master browser server MUST remove the server from the backup browser server list. It MUST also reevaluate its backup browser server list according to the algorithm (as specified in section 3.3.5.7) and it issue BecomeBackup messages to selected servers, ensuring that there are sufficient backup browser servers based on the number of computers in the machine group.

3.3.7 Other Local Events

When shutting down, the browser server MUST send a final HostAnnouncement frame with a ServerType value of 0. If the browser server is also a local master browser server, it MUST send a RequestElection frame with both **Version** and **Criterion** fields set to 0.

3.4 Domain Master Browser Details

A domain master browser server for a domain MUST act as a local master browser server for its subnet. Therefore, it acts exactly like a local master browser server (section 3.3) except where indicated differently in this section.

Historical note: By convention, the PDC of a domain SHOULD also be the domain master browser server for two reasons:

- Some browser servers make use of the fact that the PDC and domain master browser server are on the same machine. In this way, they locate the name of the PDC. For more information, see [MS-ADTS]. After the name of the machine is known, the client can form the NetBIOS name <name of the machine>[0x1D] to construct a NetBIOS name that can only be registered by the domain master browser server.
- The domain master browser registers the <domain>[0x1B] and <domain>[0x1C] records to the Windows Internet Name Service(WINS) server if the browser protocol is implemented in a domain with WINS. For more information, see [MS-ADTS] section 7.3.4. The WINS server ensures that the network transport address of the machine that registered the <domain>[0x1B] address is always the first address returned when the <domain>[0x1C] address is queried.<42>

3.4.1 Abstract Data Model

This section describes a hypothetical model of domain master browser server data organization that could be implemented to support this protocol. The purpose of this description is simply to help explain how this aspect of the protocol works. This specification does not prescribe that implementations adhere to this model as long as their external behavior is consistent with what is described throughout this document.

Local Master Browser Servers List: A list of the local master browser servers for the domain master browser server's domain on each subnet. The local master browser servers list MUST NOT contain duplicate server names.

Master List of Servers: Identical to the servers list as specified in section 3.3.1, except that it contains a list merged from all subnets in the domain.

Master List of Machine Groups: Identical to the machine groups as specified in section 3.3.1, except that it contains a list merged from all subnets in the domain.

The data model MAY also include the following:

Cross-Domain List: A configured list of domains, for which the domain master browser supports cross-domain browsing. Each entry contains the name of the domain master browser server, which MUST be dynamically discovered. For more information, see [MS-ADTS].<43>

3.4.2 Timers

A domain master browser server has the following timers, in addition to those as specified in section 3.3.2.

Domain Discovery Timer: Used to periodically discover the domain master browser server for each domain in the cross-domain list. The default value for this timer SHOULD be 15 minutes.

Local Master Browser Server Expiration Timer: For each local master browser server in the domain master browser servers list, the domain master browser server keeps an expiration timer. The default value for this timer SHOULD be 36 minutes.

3.4.3 Initialization

If a cross-domain list is configured, the domain master browser server MUST attempt to discover the domain master browser server for each domain that is configured, and MUST start the domain discovery timer. For more information, see [MS-ADTS].

A domain master browser server MUST register the NetBIOS unique name <domain>[0x1D], as well as the NetBIOS group names <domain>[0x00] and <domain>[0x1E].

3.4.4 Higher-Layer Triggered Events

None.

3.4.5 Message Processing Events and Sequencing Rule

3.4.5.1 Receiving a MasterAnnouncement Frame

When a MasterAnnouncement frame is received, the domain master browser server MUST update its local master browser servers list to add or update the entry for the sender of the frame. The expiration timer flocal master browser serveror this local master browser MUST be reset. The server MUST issue a **NetServerEnum2** request to the local master browser server that announced itself, and add or update the entries received by the Master List of Servers.

Timer Events

Domain Discovery Timer: If a domain discovery timer expires, and a cross-domain list is configured, the domain master browser server MUST attempt to rediscover the PDC for that domain to verify that the domain still exists. For more information, see [MS-ADTS].

Local Master Browser Server Expiration Timer: When the local master browser server expiration timer expires, the domain master browser server MUST remove the local master browser server that expired from the local master browser servers list.

3.4.6 Other Local Events

If a domain master browser server ceases to be a domain master browser (for example, on shutdown, or if its role is locally changed by an administrator), it MUST deregister the NetBIOS unique name <domain>[0x1D], as well as the NetBIOS group names <domain>[0x00] and <domain>[0x1E].

4 Protocol Examples

The following sections describe operations as used in common scenarios to illustrate the function of the CIFS Browser Protocol.

4.1 Mailslot Frame Example

The following is an example of a generic browser SMB ([MS-SMB]) that shows how a browser message 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, FileName = \MAILSLOT\BROWSE
  SMB: SMB Status = No Error
    SMB: ErrorClass = No Error
    SMB: Error = No Error
  SMB: Header: TID = 0x0000 PID = 0x0000 UID = 0x0000 MID = 0x0000
    SMB: Flags = 0 (0x0)
    SMB: Flags2 = 0 (0x0)
    SMB: TreeID = 0 (0x0)
    SMB: ProcessID = 0 (0x0)
    SMB: UserID = 0 (0x0)
    SMB: MultiplexID = 0 (0x0)
  SMB: Command = C transact
    SMB: WordCount = 17 (0x11)
    SMB: TotalParameterCount = 0 (0x0)
    SMB: TotalDataCount = 33 (0x21)
    SMB: MaxParameterCount = 0 (0x0)
    SMB: MaxDataCount = 0 (0x0)
    SMB: MaxSetupCount = 0 (0x0)
    SMB:  Flags = Do NOT disconnect TID
      SMB: Disconnect = ..............0 (Do NOT disconnect TID)
      SMB: Reserved  =   000000000000000. (Reserved)
    SMB: Timeout = 1000 milli sec(s)
    SMB: ParameterCount = 0 (0x0)
    SMB: ParameterOffset = 0 (0x0)
    SMB: DataCount = 33 (0x21)
    SMB: DataOffset = 86 (0x56)
    SMB: SetupCount = 3 (0x3)
    SMB: MailSlotsSetupWords
      SMB: MailSlotOpcode = Write Mail Slot
      SMB: TransactionPriority = 0 (0x0)
      SMB:  MailSlotClass = Unreliable & Broadcast
    SMB: ByteCount = 50 (0x32)
    SMB: MailSlotsBuffer:
      SMB:  FileName = \MAILSLOT\BROWSE
BROWSER: Local Master Announcement
   BROWSER: Command = Local Master Announcement, 15(0x0F)
   BROWSER: UpdateCount = 0 (0x0)
   BROWSER: Periodicity = 720000 (12 minutes)
   BROWSER: ServerName = GERMANSHA
   BROWSER: OSVersionMajor = 5 (0x5)
   BROWSER: OSVersionMinor = 1 (0x1)
   BROWSER: ServerType = 0x00051003
   BROWSER: BrowserVersionMajor = 15 (0xF)
   BROWSER: BrowserVersionMinor = 1 (0x1)
   BROWSER: Signature = 43605 (0xAA55)
```

4.2 A Browser Server Wins the First Election Round and the Election

The diagram in Figure 2 depicts the following election process.

- A browser server receives a RequestElection frame and determines that it is winning the election, as compared to the sender of the RequestElection (section 2.2.4) frame.
- The potential master browser server sends out a RequestElection frame that contains its own election **Version** and **Criteria** values.

  The browser server waits for 200 msec, 400 msec, or 800 msec, based on its role in the machine group (as specified in section 3.3.5.8), and then repeats the RequestElection frame.
- Because the browser server does not receive any RequestElection frames, it repeats the process another three times.
- Finally, the browser server declares itself a winner.
- A potential browser server (on the right) receives a RequestElection frame and decides it is winning the election, as compared to the sender of the RequestElection frame (for more information, see section 2.2.4).
- The potential browser server sends out a RequestElection frame that contains its own election **Version** and **Criteria** values.
- Meanwhile, a browser server from the browser cloud (on the left) has also received the first RequestElection frame, and decides that it is a winner.
- That browser server sends a RequestElection frame that specifies its own election **Version** and **Criteria** values.
- The potential browser server receives this new RequestElection frame and decides that it has lost the election, when it compares itself to the other browser server.
- The potential browser server on the left sends a total of four RequestElection frames, and receives no responses. It declares itself a winner.

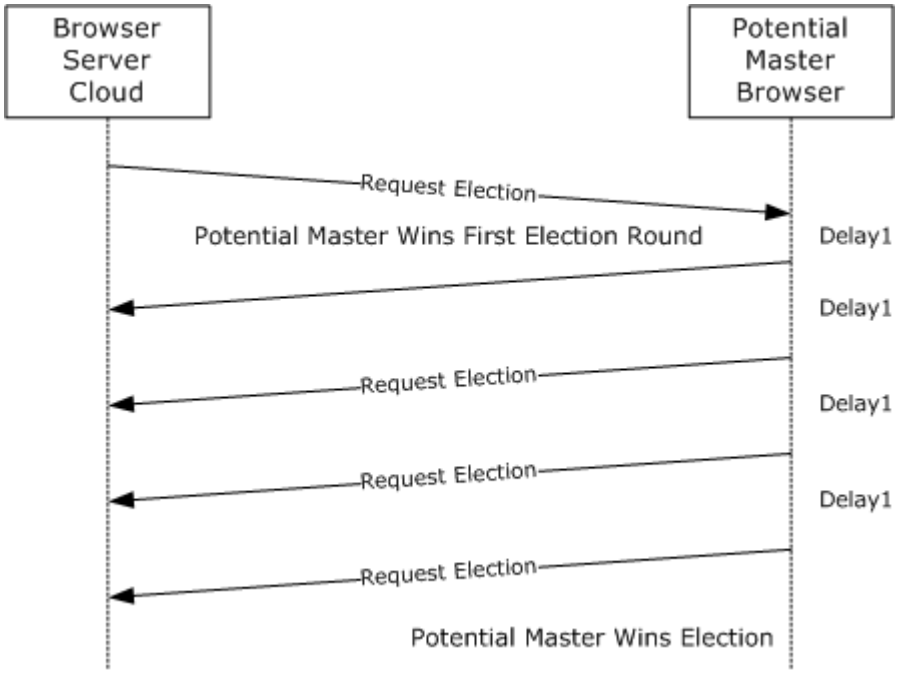


Figure 2: A browser server wins the first election round and the election

4.3 A Browser Server Wins the First Round but Loses the Election

The diagram in Figure 3 depicts the following election process.

- A potential browser server (on the right) receives a RequestElection frame and decides it is winning the election, as compared to the sender of the RequestElection frame (for more information, see section 2.2.4).
- The potential browser server sends out a RequestElection frame that contains its own election version and criterion values.
- Meanwhile, a browser server from the browser cloud (on the left) has also received the first RequestElection frame, and decides that it is a winner.
- That browser server sends a RequestElection frame that specifies its own election version and criterion values.
- The potential browser server receives this new RequestElection frame and decides that it has lost the election, when it compares itself to the other browser server.
- The potential browser server on the left sends a total of four RequestElection frames, and receives no responses. It declares itself a winner.

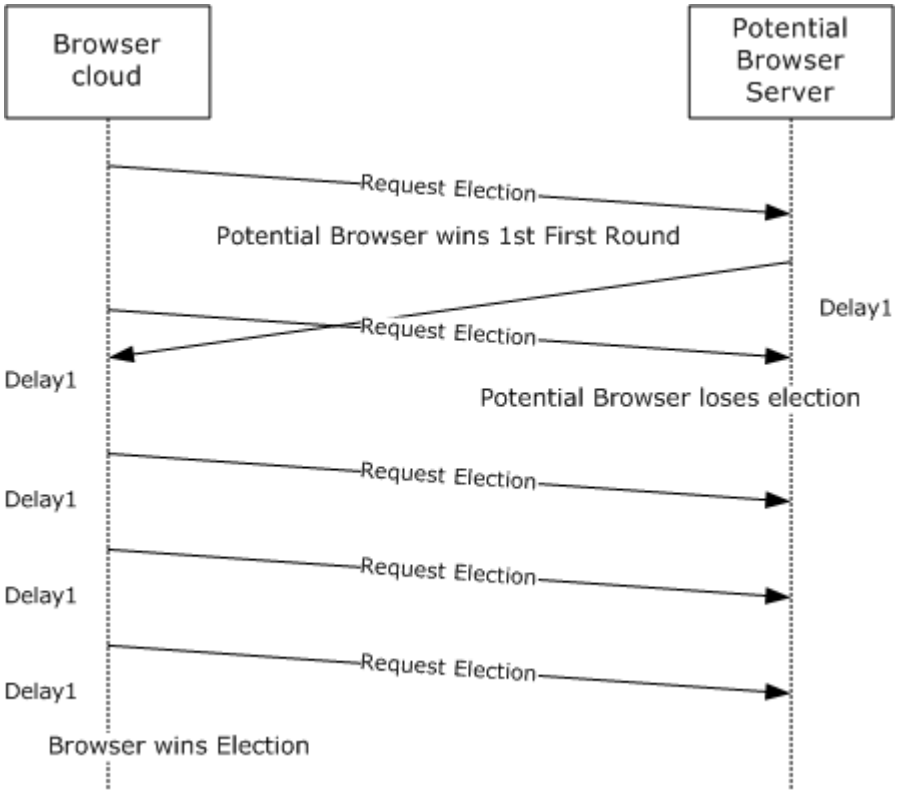


Figure 3: A browser server wins the first round but loses the election

5 Security

The following sections specify security considerations for implementers of the CIFS Browser Protocol.

5.1 Security Considerations for Implementers

In general, the browser service operates without any security. It is possible for applications to spoof elections. Additionally, malfunctioning local master browser servers can mount an effective denial of service attack against the entire browser infrastructure (for example, if a browser server refuses to release the <machine group>[0x1D] name after losing an election).

The browser service uses null sessions to establish a connection to the IPC$ share of the server. Null sessions are simply SMB connections [MS-SMB] that use no password, no domain, and no user ID to establish the connection. This implies that the connection is highly insecure.

5.2 Index of Security Parameters

This protocol has no security parameters.

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows Vista
- Windows Server 2003
- Windows XP
- Windows 2000
- Windows Me
- Windows 98

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 1.8: The following table shows the unsigned 8-bit major and minor operating system version numbers that are used by Windows clients and servers.

Operating system	Major version	Minor version
Windows 95	0x04	0x00
Windows 98	0x04	0x0A
Windows Me	0x04	0x5A
Windows NT 4.0	0x04	0x00
Windows 2000	0x05	0x00
Windows XP	0x05	0x01
Windows Server 2003	0x05	0x02
Windows Server 2003 R2	0x05	0x02
Windows Vista	0x06	0x00
Windows Server 2008	0x06	0x00

<2> Section 1.8: Windows only uses these values as specified in [MS-ERREF].

<3> Section 2.1: Windows clients always use \MAILSLOT\BROWSE for mailslot requests.

<4> Section 2.1.1.2: Windows–based servers attempt to register the machine group name three times; if all attempts fail, it can be concluded that another server is already the master browser server for this domain. Name registration is as specified in [RFC1001] section 5.2.

<5> Section 2.1.1.2: Windows Internet Name Service (WINS) keeps up to 25 IP addresses for the <domain>[0x1C] group name. The PDC is the only machine that registers the <domain>[0x1B] name with WINS. WINS ensures that the IP address corresponding to the computer that registered

<domain>[0x1B] is always placed first in this list of up to 25 IP addresses that registered <domain>[0x1C].

<6> Section 2.2.2: The Windows announcement frequency is as specified in section 3.2.6.

<7> Section 2.2.2: For more information, see section 1.8 for Windows operating system values.

<8> Section 2.2.2: For more information, see section 1.8 for Windows operating system values.

<9> Section 2.2.2: The comment field in Windows is a single byte containing the value 0x00.

<10> Section 2.2.4: The 0x00000002 role value is set when the *MaintainServerList* parameter is set for the workstation service.

<11> Section 2.2.4: The 0x00000002 role value is set when the *MaintainServerList* parameter is set for the workstation service.

<12> Section 2.2.5: The default value on Windows is 0x04.

<13> Section 2.2.5: The value chosen by the Windows client for Token is 0x00000001.

<14> Section 2.2.8: Windows clients and servers set BrowserConfigVersionMajor to 0x03.

<15> Section 2.2.8: Windows clients and servers set BrowserConfigVersionMinor to 0x0A.

<16> Section 2.2.8: Windows servers leave this field uninitialized, so the value is undefined.

<17> Section 2.2.8: Windows servers leave this field uninitialized, so the value is undefined.

<18> Section 2.2.11: For more information on Windows operating system values, see section 1.8.

<19> Section 2.2.11: For more information about Windows operating system values, see section 1.8.

<20> Section 2.2.11: Windows clients and servers set BrowserConfigVersionMajor to 0x03.

<21> Section 2.2.11: Windows clients and servers set BrowserConfigVersionMinor to 0x0A.

<22> Section 3.1.3: Windows chooses as many servers as are returned by the local master browser server, up to a limit of three servers from the list, and uses a pseudo-random number generator when shuffling the server list.

<23> Section 3.1.3: For Windows machines, the requests to enumerate machines on the network would also fail if the initialization has failed.

<24> Section 3.1.5.1.1: The browser client uses an initial token value of 0, and it increments this value every time it makes a GetBackupListRequest.

<25> Section 3.1.5.1.2: The browser client selects up to three of the backup browser servers and uses a pseudo-random number generator to determine which one of the servers to use from that list.

<26> Section 3.1.6: Upon expiration of the timer, a Windows–based client retransmits the GetBackupListRequest twice more and, if both transmissions result in no GetBackupListResponse frames being received, a Windows–based client sends a RequestElection frame (section 2.2.4) with the version and criteria values set to 0. For more information about the election process, see section 3.3.5.8.

<27> Section 3.1.6: All versions of Windows use the default retry count of 3 before sending the RequestElection frame.

<28> Section 3.2.6: Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008 use the time-out values as specified in this section. Windows 98 and Windows 2000 use the following time-out values.

Server.HostAnnouncementCount value (as specified in section 3.2.1)	New HostAnnouncement timer value
1	4 minutes
2	8 minutes
> 2	12 minutes

<29> Section 3.3: The new local master browser server sends the announcement request only if its server list is empty. If the local master browser server was previously a backup browser, it does not send the announcement request.

<30> Section 3.3.1: The Machine Groups list is ordered alphabetically.

<31> Section 3.3.1: In Windows, the servers list is ordered alphabetically.

<32> Section 3.3.2: The local master browser will use the timeout values as specified in the DomainAnnouncement table in section 3.3.6.

<33> Section 3.3.2: The local master browser uses the default timeout values as specified in the LocalMasterAnnouncement table in section 3.3.6.

<34> Section 3.3.2: The local master browser will use the default timeout of 5 minutes.

<35> Section 3.3.2: Windows implementations use the default value of 5 minutes for this timer.

<36> Section 3.3.5: Malformed messages are written to the Windows event log.

<37> Section 3.3.5.1: If the FindMaster timer expires six times (9 seconds), the browser server sends a RequestElection (section 2.2.4) frame.

<38> Section 3.3.5.6: The Windows browser server issues the NetServerEnum2/NetServerEnum3 request if it can determine the local master browser for the remote domain. If it cannot determine the name of the local master browser for the remote domain, it fails the request with error NERR_DevNotRedirected (0x0000083B).

<39> Section 3.3.5.7: A Windows local master browser attempts to maintain a number of backup browser servers that it nominates as follows.

Number of servers	Number of backup browser servers
1	0
2-31	1
32-63	2
> 63	3

<40> Section 3.3.6: Windows XP, Windows Server 2003, and Windows Vista use the time-out values listed above. Windows 98, Windows Me, and Windows 2000 use the following time-out values.

LocalMasterAnnouncement timer count value	New LocalMasterAnnouncement timer value
1	4 minutes
2	8 minutes
> 2	12 minutes

<41> Section 3.3.6: Servers lists and machine groups lists are merged alphabetically.

<42> Section 3.4: WINS maintains a set of up to 25 addresses for the <domain>[0x1C] group address.

<43> Section 3.4.1: Windows supports a cross-domain list.

7 Index

N

O

P

R

S

T

U

V

W