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

This protocol documentation is intended for use in conjunction with publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

A protocol specification does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them.

Revision Summary

Date	Revision History	Revision Class	Comments
09/28/2007	0.1	Major	MCPP Milestone M5+90 Initial Availability
10/23/2007	1.0	Major	Updated and revised the technical content.
11/30/2007	1.0.1	Editorial	Revised and edited the technical content.
01/25/2008	1.0.2	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction

This document specifies the Common Internet File System (CIFS) Browser Auxiliary Protocol Specification. This protocol is used by the **master browser server** and **Domain Master Browser server** as defined in [MS-BRWS]. The master browser server uses this protocol to query configuration information for the **domains** from the domain master browser server. The protocol operation is stateless.

1.1 Glossary

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

Browser
Browser Server
Client
Domain
Endpoint
Handle
Interface Definition Language (IDL)
Master Browser Server
Named Pipe
Opnum
Primary Domain Controller (PDC)
Server
Unicode
Universal Unique Identifier (UUID)

The following terms are specific to this document:

Domain Master Browser Server: A **master browser server** that is responsible for combining information for an entire **domain** across all **subnets**. A **domain master browser server** is responsible for keeping multiple **subnets** in synchronization by periodically querying the **local master browser server** for information about user accounts, security, and available resources, such as printers.

Master Browser Server: A **browser server** that is responsible for maintaining a master list of available resources on a **subnet**. Each **subnet** requires a **master browser server**. The **master browser server** for a particular **domain** is called the **domain master browser server** For more information about **master browser servers**, see [MS-BRWS].

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

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

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site,

http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

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

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

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

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

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions", January 2007.

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

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

[RFC1123] Braden, R., "Requirements for Internet Hosts–Application and Support", RFC 1123, October 1989, http://www.ietf.org/rfc/rfc1123.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-BRWS] Microsoft Corporation, "Common Internet File System (CIFS) Browser Protocol Specification", July 2007.

[MSBRLIST] Microsoft Corporation, "How to support browse lists for multiple domains", January 2007, http://support.microsoft.com/kb/191388

[PIPE] Microsoft Corporation, "Named Pipes", http://msdn2.microsoft.com/en-us/library/aa365590.aspx

1.3 Protocol Overview (Synopsis)

The main objective of the CIFS Browser Auxiliary Protocol is to provide a method for the master browser server of a subnet to query specific additional information from the domain master browser server for a given domain. Selection of the master browser server and domain master browser and the roles that these **servers** play are as specified in [MS-BRWS].

1.4 Relationship to Other Protocols

This protocol depends on RPC, as specified in [MS-RPCE], for its transport. This protocol uses RPC over **named pipes**, as specified in [MS-RPCE] section 2.1.1.2. Named pipes use the Server Message Block (SMB) Protocol, as specified in [MS-SMB].

An implementation of [MS-BRWS] may use this protocol to retrieve information from the domain master browser.

1.5 Prerequisites/Preconditions

The master browser server has previously identified the **endpoint** address of the domain master browser server.

1.6 Applicability Statement

This protocol is used to retrieve the list of domains that the domain master browser server has been configured to support.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

Parameter	Value	Reference
RPC Interface UUID	6BFFD098-A112-3610-9833-012892020162	See section 2.1
Pipe Name	"\pipe\browser"	See section 2.1

2 Messages

2.1 Transport

The RPC methods that the CIFS Browser Auxiliary Protocol uses are available on one endpoint:

- "\pipe\browser" named pipe (RPC protseqs ncacn_np), as specified in [MS-RPCE] section 2.1.1.2.

The CIFS Browser Auxiliary Protocol endpoint is available only over named pipes. For more information about named pipes, see [PIPE].

This protocol MUST use the **universally unique identifier (UUID)** as specified in section 1.9. The RPC version number is 3.0.

This protocol allows any user to establish a connection to the RPC server. The protocol uses the underlying RPC protocol to retrieve the identity of the caller that made the method call, as described in section 3.3.4.3 of [MS-RPCE]. The server SHOULD use this identity to perform method specific access checks as described in section 3.1.5.

2.2 Common Data Types

In addition to RPC base types and definitions specified in [C706] and [MS-RPCE], additional data types are defined below.

The following are the types that are defined in this specification.

2.2.1 Simple Data Types

2.2.1.1 BROWSER_IDENTIFY_HANDLE

The **BROWSER_IDENTIFY_HANDLE** structure is a null-terminated **Unicode** string that identifies the remote computer on which to execute the method.

This type is declared as follows:

```
typedef [handle] LPTSTR BROWSER_IDENTIFY_HANDLE;
```

The **client** MUST set the impersonation level for the RPC connection that refers to this **handle** to "IDENTIFICATION". "IDENTIFICATION" implies an impersonation level of SECURITY_IDENTIFICATION. For more information on impersonation levels, see the **ImpersonationLevel** field in [MS-SMB] section 2.2.8.

2.2.2 Constants

2.2.2.1 Platform IDs

The following values specify the information level to use for **platform**-specific information on the server.<1>

Name	Value (decimal)
PLATFORM_ID_DOS	300
PLATFORM_ID_OS2	400
PLATFORM_ID_NT	500
PLATFORM_ID_OSF	600
PLATFORM_ID_VMS	700

2.2.3 Structures

2.2.3.1 SERVER_INFO_100

The **SERVER_INFO_100** structure contains information about the specified server, including the name and platform.

```
typedef struct _SERVER_INFO_100 {
  DWORD sv100_platform_id;
  [string] wchar_t* sv100_name;
} SERVER_INFO_100,
 *PSERVER_INFO_100,
 *LPSERVER_INFO_100;
```

sv100_platform_id: Specifies the information level to use for platform-specific information. This member MUST be one of the values that is listed in section 2.2.2.1.

sv100_name: A pointer to a null-terminated Unicode UTF-16 string specifying the DNS name (as specified in [RFC1123] section 2.1) or NetBIOS name (as specified in [RFC1001] section 14 and [RFC1002] section 4) of a server.

2.2.3.2 SERVER_INFO_100_CONTAINER

The **SERVER_INFO_100_CONTAINER** structure contains a count of the entries returned by the method and a pointer to a buffer.

```
typedef struct _SERVER_INFO_100_CONTAINER {
  DWORD EntriesRead;
  [size_is(EntriesRead)] LPSERVER_INFO_100 Buffer;
} SERVER_INFO_100_CONTAINER,
 *PSERVER_INFO_100_CONTAINER,
 *LPSERVER_INFO_100_CONTAINER;
```

EntriesRead: The number of entries returned by the method call. This value MUST be zero if no domains are configured in the **Primary Domain Controller**.

Buffer: A pointer to an array of **SERVER_INFO_100** data structures. If EntriesRead is zero, this field is undefined and MUST NOT be considered a valid pointer.

2.2.3.3 SERVER_ENUM_STRUCT

The **SERVER_ENUM_STRUCT** structure defines the layout for a structure with a value to indicate the information level submitted to the method and a pointer to a data structure that contains an array of data structures returned by the method. This structure is used by I_BrowserrQueryOtherDomains.

```
typedef struct _SERVER_ENUM_STRUCT {
  DWORD Level;
  [switch_is(Level)] union {
    [case(100)]
      LPSERVER_INFO_100_CONTAINER Level100;
  } ServerInfo;
} SERVER_ENUM_STRUCT,
 *PSERVER_ENUM_STRUCT,
 *LPSERVER_ENUM_STRUCT;
```

Level: The information level of the data. This member MUST be 100.

ServerInfo: A structure that contains an array of data structures. The **Level** member determines the data type of the members of this array.

Level100: A pointer to a **SERVER_INFO_100_CONTAINER** structure that contains the number of entries returned by the method and a pointer to an array of **SERVER_INFO_100** structures.

3 Protocol Details

3.1 Server Details

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

Section 2.1 specifies the parameters necessary to initialize the RPC protocol.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

The *ServerName* parameter MUST be ignored by the server when processing any message.<2>

3.1.5.1 Browser

The **Browser** interface lists the methods associated with the **Browser** service, which creates and maintains a view of resources available on a network. The server does not maintain client state information. The protocol operation is stateless. The version for this interface is 0.0.

The UUID for this interface is: "6BFFD098-A112-3610-9833-012892020162".

Methods in RPC Opnum Order

Method	Description
Opnum0NotUsedOnWire	Reserved for local use. Opnum: 0
Opnum1NotUsedOnWire	Reserved for local use. Opnum: 1
I_BrowserrQueryOtherDomains	Returns a list of other domains configured for this computer. Opnum: 2
Opnum3NotUsedOnWire	Reserved for local use. Opnum: 3
Opnum4NotUsedOnWire	Reserved for local use. Opnum: 4
Opnum5NotUsedOnWire	Reserved for local use. Opnum: 5

Method	Description
Opnum6NotUsedOnWire	Reserved for local use. Opnum: 6
Opnum7NotUsedOnWire	Reserved for local use. Opnum: 7
Opnum8NotUsedOnWire	Reserved for local use. Opnum: 8
Opnum9NotUsedOnWire	Reserved for local use. Opnum: 9
Opnum10NotUsedOnWire	Reserved for local use. Opnum: 10
Opnum11NotUsedOnWire	Reserved for local use. Opnum: 11

In the preceding table, the phrase "Reserved for local use" means that the client MUST NOT send the **opnum** and that the server behavior is undefined<3> because it does not affect interoperability.

3.1.5.1.1 I_BrowserrQueryOtherDomains (Opnum 2)

The **I_BrowserrQueryOtherDomains** method is received by the server in an RPC_REQUEST packet. In response, the server MUST retrieve the names of all domains configured for this computer.<4> The server MUST be a Primary Domain Controller (PDC) that is acting as the **Domain Master Browser Server** for this method to execute successfully. If this server is not a Primary Domain Controller it MUST fail the request.

```
NET_API_STATUS I_BrowserrQueryOtherDomains(
  [in, string, unique] BROWSER_IDENTIFY_HANDLE ServerName,
  [in, out] LPSERVER_ENUM_STRUCT InfoStruct,
  [out] LPDWORD TotalEntries
);
```

ServerName: An optional **BROWSER_IDENTIFY_HANDLE** structure that specifies the name of the server to execute the method. This value is ignored upon receipt.

InfoStruct: A pointer to a **SERVER_ENUM_STRUCT** structure that contains the information level parameter and a pointer to a SERVER_INFO_x structure, where <x> MUST be 100. The *Level* parameter MUST be set to 100. If *Level* is set to any other value, the method MUST return ERROR_INVALID_LEVEL. If the method return value is neither NERR_Success nor ERROR_MORE_DATA, the return value of InfoStruct is NULL.

TotalEntries: The number of entries returned by the method call. This field MUST match the **EntriesRead** field of the **SERVER_INFO_100_CONTAINER** structure.

Return Values: The method returns 0 (NERR_Success) on success; otherwise, it returns a nonzero error code, as specified in either Win32 Error Codes. The most common error codes are listed in the following table.<5>

Return value/code	Description
0x00000000 NERR_Success	The operation completed successfully.
0x00000005 ERROR_ACCESS_DENIED	Access is denied.
0x00000008 ERROR_NOT_ENOUGH_MEMORY	This value MUST be returned when the server could not allocate enough memory to complete this operation.
0x00000057 ERROR_INVALID_PARAMETER	This value MUST be returned when a parameter is incorrect. For example, this value is returned when InfoStruct is NULL or the **Level100** member in the structure pointed to by InfoStruct is NULL.
0x0000007C ERROR_INVALID_LEVEL	This value MUST be returned when the **Level100** member in the structure pointed to by InfoStruct is not 100.
0x000000EA ERROR_MORE_DATA	More data is available.
0x000008B2 NERR_NotPrimary	This is not the domain master browser server.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Client Details

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

The client MUST create an RPC connection to the remote computer by using details as specified in section 2.1.

3.2.4 Message Processing Events and Sequencing Rules

There are no events or sequencing rules.

3.2.5 Timer Events

None.

3.2.6 Other Local Events

None.

4 Protocol Examples

The method provided by this protocol is a simple request-response. The server receives the request, executes the method, and returns a completion. The client simply returns the completion status to the calling application. This is a stateless protocol; each method call is independent of any previous method calls.

For example, the client calls the **I_BrowserrQueryOtherDomains** method; on receiving this method, the server executes the call locally and returns the appropriate information and NERR_Success.

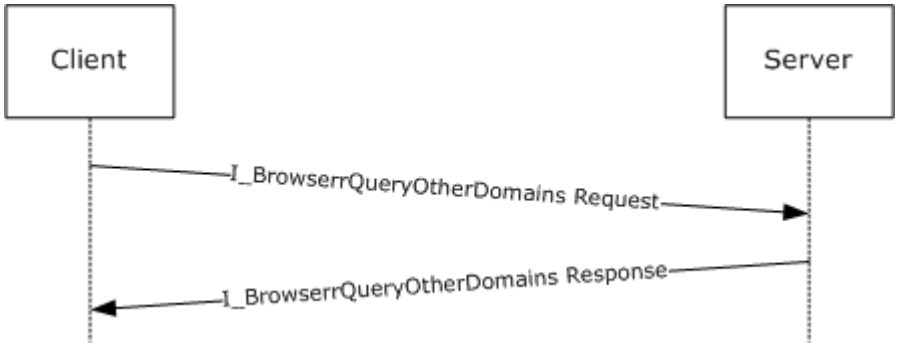

Figure 1: Simple request-response model

5 Security

5.1 Security Considerations for Implementers

As described in section 2.1, this protocol allows any user to connect to the server. Therefore, any security bug in the server implementation could be exploitable. The server implementation should enforce security on each method.

5.2 Index of Security Parameters

This protocol has no security parameters.

6 Appendix A: Full IDL

For ease of implementation, the full **IDL** is provided below, where "ms-dtyp.idl" refers to the IDL found in [MS-DTYP] Appendix A.

The syntax uses the IDL syntax extensions defined in [MS-RPCE] sections 2.2.4 and 3.1.5.1. For example, as noted in [MS-RPCE] section 2.2.4.8, a pointer_default declaration is not required and pointer_default(unique) is assumed.

```
[
    uuid(6BFFD098-A112-3610-9833-012892020162),
    version(0.0),
    ms_union,
    pointer_default(unique)
]
interface browser
{
    import "ms-dtyp.idl";

    typedef [handle] LPTSTR BROWSER_IDENTIFY_HANDLE;

    typedef struct _SERVER_INFO_100 {
        DWORD sv100_platform_id;
        [string] wchar_t* sv100_name;
    } SERVER_INFO_100,
     *PSERVER_INFO_100,
     *LPSERVER_INFO_100;

   typedef struct _SERVER_INFO_100_CONTAINER {
        DWORD   EntriesRead;
        [size_is(EntriesRead)] LPSERVER_INFO_100 Buffer;
     } SERVER_INFO_100_CONTAINER,
      *PSERVER_INFO_100_CONTAINER,
      *LPSERVER_INFO_100_CONTAINER;

    typedef struct _SERVER_ENUM_STRUCT {
        DWORD   Level;
        [switch_is(Level)] union _SERVER_ENUM_UNION {
            [case(100)]
                LPSERVER_INFO_100_CONTAINER Level100;
            [default]
                ;
        } ServerInfo;
    } SERVER_ENUM_STRUCT,
     *PSERVER_ENUM_STRUCT,
     *LPSERVER_ENUM_STRUCT;


    NET_API_STATUS Opnum0NotUsedOnWire(void);

    NET_API_STATUS Opnum1NotUsedOnWire(void);

    NET_API_STATUS
    I_BrowserrQueryOtherDomains(
        [in,string,unique] BROWSER_IDENTIFY_HANDLE ServerName,
```

```
        [in,out]            LPSERVER_ENUM_STRUCT    InfoStruct,
        [out]               LPDWORD                 TotalEntries
    );

    NET_API_STATUS Opnum3NotUsedOnWire(void);

    NET_API_STATUS Opnum4NotUsedOnWire(void);

    NET_API_STATUS Opnum5NotUsedOnWire(void);

    NET_API_STATUS Opnum6NotUsedOnWire(void);

    NET_API_STATUS Opnum7NotUsedOnWire(void);

    NET_API_STATUS Opnum8NotUsedOnWire(void);

    NET_API_STATUS Opnum9NotUsedOnWire(void);

    NET_API_STATUS Opnum10NotUsedOnWire(void);

    NET_API_STATUS Opnum11NotUsedOnWire(void);
}
```

7 Appendix B: Windows Behavior

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

- Windows Vista
- Windows Server 2003
- Windows XP
- Windows 2000

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 2.2.2.1: PLATFORM_ID_NT should be used for Windows NT Server, Windows 2000, Windows XP, Windows Server 2003, Windows Vista and Windows Server 2008.

<2> Section 3.1.5: Windows implementation uses the RPC protocol to retrieve the identity of the caller as described in section 3.3.4.3 of [MS-RPCE]. The server uses the underlying Windows security subsystem to determine the permissions for the caller. If the caller does not have the required permissions to execute a specific method, the method call fails with ERROR_ACCESS_DENIED (0x00000005).

<3> Section 3.1.5.1: Opnums reserved for local use apply to Windows as follows:

opnum	Description
0,5,6,10,11	Only used locally by Windows, never remotely.
1,3,4,9	Just returns ERROR_NOT_SUPPORTED. It is never used.
7,8	Just returns NERR_Success. It is never used.

<4> Section 3.1.5.1.1: The names of other domains configured are stored in the registry for Windows–based Primary Domain Controllers. By default no other domains are configured in the registry. These names are read to memory from the registry at startup, and are changed as the status of domains changes.

<5> Section 3.1.5.1.1: If the caller is not an authenticated user, then the method call fails with ERROR_ACCESS_DENIED (0x00000005).

8 Index

T

V

W