[MS-OXABREF]:
Address Book Name Service Provider Interface (NSPI) Referral Protocol Specification

---

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
04/25/2008	0.2		Revised and updated property names and other technical content.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Revised and edited technical content.
09/03/2008	1.02		Updated references.
12/03/2008	1.03		Updated IP notice.
04/10/2009	2.0		Updated technical content and applicable product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	4.0.0	Major	Updated and revised the technical content.
02/10/2010	4.1.0	Minor	Updated the technical content.
05/05/2010	4.2.0	Minor	Updated the technical content.
08/04/2010	4.3	Minor	Clarified the meaning of the technical content.
11/03/2010	5.0	Major	Significantly changed the technical content.
03/18/2011	6.0	Major	Significantly changed the technical content.
08/05/2011	6.0	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

The Address Book Name Service Provider Interface (NSPI) Referral Protocol defines a **remote procedure call (RPC)** service that supplies a caller with the name of an **NSPI** server. Additionally, this protocol can return the **Domain Name System (DNS) fully qualified domain name (FQDN)** of a **mailbox** server, given the **distinguished name (DN)** of that server.

Sections 1.8, 2, and 3 of this specification are normative and contain RFC 2119 language. Sections 1.5 and 1.9 are also normative but cannot contain RFC 2119 language. All other sections and examples in this specification are informative.

1.1 Glossary

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

distinguished name (DN)
Domain Name System (DNS)
dynamic endpoint
flags
fully qualified domain name (FQDN)
Interface Definition Language (IDL)
Kerberos
name service provider interface (NSPI)
Network Data Representation (NDR)
NT LAN Manager (NTLM) Authentication Protocol
opnum
remote procedure call (RPC)
RPC protocol sequence
universally unique identifier (UUID)
well-known endpoint

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

Address Book object
endpoint
mailbox
public folder

The following terms are specific to this document:

binding handle: A data structure that represents the logical connection between a client and a server.

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-ERREF] Microsoft Corporation, "Windows Error Codes".

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

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

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

1.2.2 Informative References

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

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

[MS-OXGLOS] Microsoft Corporation, "Exchange Server Protocols Master Glossary".

[MS-OXOABK] Microsoft Corporation, "Address Book Object Protocol Specification".

1.3 Overview

This protocol enables clients to retrieve the network name of a server from a name service provider interface (NSPI) referral server. Clients use this protocol before performing any NSPI requests, in order to retrieve the name of the NSPI server to connect to. This gives the NSPI referral server the ability to control which NSPI server an NSPI client will connect to, for purposes including but not limited to balancing the client load across multiple NSPI servers, choosing the best version of NSPI server for that particular client, or satisfying network requirements that are not discernible by the client. Clients also use this protocol to retrieve the fully qualified domain name (FQDN) of the NSPI server, when only the distinguished name (DN) the mailbox server is known. Figure 1 shows the request to the NSPI referral server for the name of the NSPI server and the server's response to the client. Figure 2 shows the request to the NSPI referral server for the FQDN of the mailbox server and the server's response to the client.

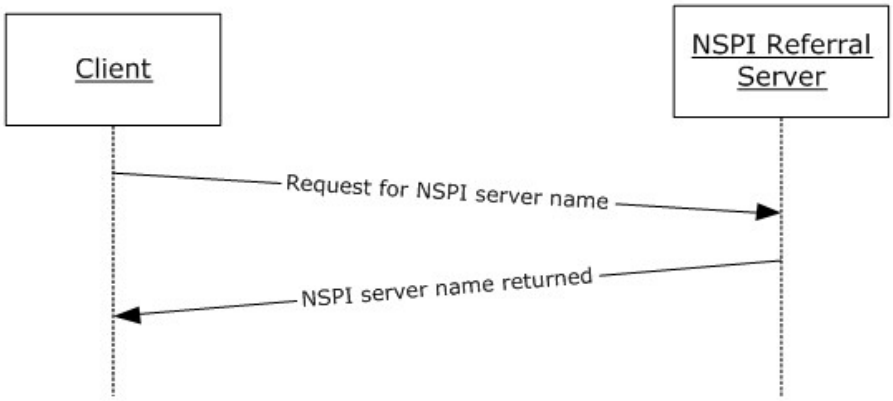

Figure 1: Client retrieving NSPI server name from the NSPI referral server

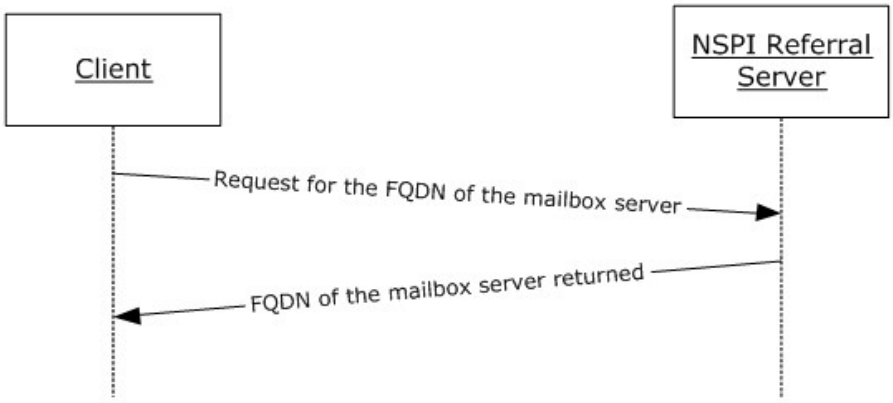

Figure 2: Client retrieving mailbox server name from the NSPI referral server

1.4 Relationship to Other Protocols

This protocol is built on the remote procedure call (RPC) interface, as described in [C706] and [MS-RPCE]. It supports only **RPC protocol sequences** ncacn_ip_tcp and ncacn_http, as described in [MS-RPCE].

1.5 Prerequisites/Preconditions

None.

1.6 Applicability Statement

This protocol is designed to return the name of an name service provider interface (NSPI) server before the client engages in any NSPI requests. It is also designed to return the fully qualified domain name (FQDN) of a mailbox server, as described in [RFC1035], when a client only knows the

distinguished name (DN) of a mailbox server with which it can make a network connection. In practice, this is necessary in several cases:

- When creating client mail settings, a client uses an NSPI server to read an **Address Book object** representing its mailbox, which includes the **DN** of the messaging server that hosts the mailbox.
- When connecting to the wrong mailbox or **public folder** server, an error will be returned containing the DN of the correct server.
- When connecting to another user's mailbox, having only the **PidTagAddressBookHomeMessageDatabase** property ([MS-OXOABK] section 2.2.4.67) for that mailbox.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Supported Transports:** This protocol uses multiple RPC protocol sequences, as specified in section 2.1.
- **Protocol Versions:** This protocol has only one interface version, but that interface has been extended by appending methods at the end. The use of these methods is specified in section 3.1.
- **Security and Authentication Methods:** This protocol supports the following authentication methods: the **NT LAN Manager (NTLM) Authentication Protocol** and **Kerberos**. These authentication methods are described in section 2.1.

1.8 Vendor-Extensible Fields

This protocol uses HRESULT values as specified in [MS-ERREF]. Vendors can define their own HRESULT values, provided they set the C bit (0x20000000) for each vendor-defined value, indicating the value is a customer code.

The **RfrGetNewDSA** method can also return other error values. Any nonzero return code indicates an error.

1.9 Standards Assignments

This protocol uses a **well-known endpoint**, as described in section 2.1. This protocol uses remote procedure call (RPC) **dynamic endpoints** as described in [C706] part 4.

Parameter	Value	Reference
RFRI RPC interface **universally unique identifier (UUID)**	1544f5e0-613c-11d1-93df-00c04fd7bd09	Appendix A

2 Messages

2.1 Transport

This protocol works over the following remote procedure call (RPC) sequences:

- ncacn_ip_tcp — RPC over TCP
- ncacn_http — RPC over HTTP

This protocol uses a well-known endpoint, 6002, for the RPC protocol sequence ncacn_http. For ncacn_ip_tcp, this protocol uses a dynamic endpoint, as specified in [C706] part 4. A dynamic endpoint is configurable by the owner of the server to enable the owner to open a minimum set of ports in a firewall to give clients access to the server.

The following table summarizes the **endpoints** used by this protocol.

RPC Protocol Sequence	Endpoint
ncacn_http	6002 (well-known endpoint)
ncacn_ip_tcp	dynamic endpoint

This protocol supports the NT LAN Manager (NTLM) Authentication Protocol (RPC_C_AUTHN_WINNT), and the Negotiate (RPC_C_AUTHN_GSS_NEGOTIATE) security providers. A Negotiate security provider determines whether to use NTLM or Kerberos authentication. The default is Kerberos. A Negotiate security provider selects NTLM authentication only in the following cases:

- One of the systems that is involved in the authentication cannot use Kerberos authentication.
- The client does not provide sufficient information to use Kerberos authentication.

Callers MUST be authenticated but no further authorization checks are performed.

2.2 Common Data Types

This protocol MUST indicate to the remote procedure call (RPC) runtime that it is to support the **Network Data Representation (NDR)** transfer syntax only, as specified in [C706] part 4.

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

2.2.1 handle_t

The **handle_t** data type is used to represent an explicit remote procedure call (RPC) **binding handle**, as specified in [C706] and [MS-RPCE]. It is a primitive type of the **Interface Definition Language (IDL)** and does not require an explicit declaration.

3 Protocol Details

The client side of this protocol is simply a pass-through. That is, no additional timers or other state is required on the client side of this protocol. Calls made by the higher-layer protocol or application are passed directly to the transport, and the results returned by the transport are passed directly back to the higher-layer protocol or application.

3.1 NSPI Referral Server Details

This is a simple single-request, single-response protocol.

3.1.1 Abstract Data Model

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

A data structure that tracks the available NSPI servers and their current state is beneficial to any implementation of this protocol. Tracking this internal state means the client is more likely to get a valid NSPI server name and connect successfully on the first try. The NSPI referral server is not required to connect to the NSPI server in order to service clients; therefore, it is important for an implementation of this protocol to use some method to maintain up-to-date information about available NSPI servers. This ensures that clients who call the **RfrGetNewDSA** method are not given the name of an NSPI server that is not functioning.

3.1.2 Timers

None.

3.1.3 Initialization

Initialization occurs at server startup. This protocol registers the protocol interface with the remote procedure call (RPC) system using the RFRI RPC interface universally unique identifier (UUID), from section 1.9.

3.1.4 Message Processing Events and Sequencing Rules

This protocol MUST indicate to the RPC runtime that it is to perform a strict NDR data consistency check at target level 5.0, as specified in [MS-RPCE] section 3.

This interface includes the following methods:

Method	Description
RfrGetNewDSA	Returns the name of an NSPI server. **opnum**: 0
RfrGetFQDNFromServerDN	Returns the Domain Name System (DNS) fully qualified domain name (FQDN) of the server corresponding to the passed distinguished name (DN). For more details about DNS, see [RFC1035]. opnum: 1

All methods MUST NOT throw exceptions.

3.1.4.1 RfrGetNewDSA (opnum 0)

The **RfrGetNewDSA** method returns the name of an NSPI server.

```
//opnum 0
long RfrGetNewDSA(
 [in]                          handle_t          hRpc,
 [in]                          unsigned long     ulFlags,
 [in, string]                  unsigned char *   pUserDN,
 [in,out,unique, string]       unsigned char * * ppszUnused,
 [in,out,unique, string]       unsigned char * * ppszServer);
```

hRpc: A remote procedure call (RPC) binding handle parameter, as specified in [C706] section 2. MUST NOT be NULL.

ulFlags: An unsigned long value, containing a set of bit **flags**. Unused; SHOULD be set to zero. Other values MUST be ignored by the server.

pUserDN: Optional, a distinguished name (DN) indicating the mailbox owned by the client user. The client SHOULD pass this to the server. If supplied, the server SHOULD use that DN to affect which NSPI server is returned to the caller.

ppszUnused: A string. Unused; SHOULD be set to NULL. Other values MUST be ignored by the server.

ppszServer: A string. If the server does not return an error, ppszServer contains the fully qualified domain name (FQDN) of an NSPI server. On failure, the value is undefined.

Return Values: The server returns 0 for a successful execution. An error results in an HRESULT or other nonzero error code.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol as specified in [MS-RPCE].

Upon receiving this message, the server MUST process the data from the client using the following constraints. If **pUserDN** is present and contains the DN of an Address Book object, the server MUST prioritize an NSPI server that contains a writeable copy of that Address Book object over NSPI servers that do not, and return a client access array or a server from the user's mailbox site.<1> The server can take other constraints into account, such as the network location of the NSPI server in comparison to the NSPI referral server or the client. The server MUST prioritize available, responsive NSPI servers over unresponsive ones. The server can consider load balancing of clients when more than one NSPI server has equal priority. After considering these constraints, method SHOULD return one NSPI server name in the **ppszServer** parameter and a return value of zero. If any errors occur and the method is not able to return the name of an NSPI server, a nonzero value MUST be returned.

Because the goal of the server is to balance load across multiple NSPI servers, clients MUST NOT expect the same NSPI server to be returned from the **RfrGetNewDSA** method, even if all inputs are the same.

A client SHOULD call the **RfrGetNewDSA** method and connect to the NSPI server returned from that method. The client SHOULD NOT connect to an NSPI server without first requesting a server name from **RfrGetNewDSA**.<2>

The NSPI server returned in the **ppszServer** parameter MUST support the same RPC protocol sequence used by the RPC binding handle.

3.1.4.2 RfrGetFQDNFromServerDN (opnum 1)

The **RfrGetFQDNFromServerDN** method returns the Domain Name System (DNS) fully qualified domain name (FQDN) of the server corresponding to the passed distinguished name (DN).

[C++]

```
// opnum 1
long RfrGetFQDNFromServerDN(
   [in]                                    handle_t           hRpc,
   [in]                                    unsigned long      ulFlags,
   [in, range(10,1024)]                    unsigned long      cbMailboxServerDN,
   [in, string, size_is(cbMailboxServerDN)] unsigned char   *  szMailboxServerDN,
   [out,ref,string]                        unsigned char   ** ppszServerFQDN);
```

hRpc: A remote procedure call (RPC) binding handle parameter, as specified in [C706] section 2. MUST NOT be NULL.

ulFlags: An unsigned long value, containing a set of bit flags. Unused; SHOULD be set to zero. Other values MUST be ignored by the server.

cbMailboxServerDN: An unsigned long value containing the number of bytes in the value of the **szMailboxServerDN** parameter, including terminating NULL character. The value is at least 10, at most 1024.

szMailboxServerDN: A 5 or 6-element DN identifying a mailbox server, which MUST match the server's implementation of server identities. It follows this format:

```
"/o=" organization-name "/ou=" administrative-group-name "/CN=configuration/CN=servers/CN="
instance-name "/CN=" short-messaging-server-name
```

The `CN=" instance-name "` element is optional.<3>

Note that the client MAY receive a DN identifying a specific database on this server, from sources listed in section 1.6. This DN follows this format:

```
"/o=" organization-name "/ou=" administrative-group-name "/CN=configuration/CN=servers/CN="
instance-name "/CN=" short-messaging-server-name "/CN=Microsoft Private MDB"
```

Or

```
"/o=" organization-name "/ou=" administrative-group-name "/CN=configuration/CN=servers/CN="
instance-name "/CN=" short-messaging-server-name "/CN=Microsoft Public MDB"
```

If this is the DN available, it is the client's responsibility to remove the final element before passing the DN to the **RfrGetFQDNFromServerDN** method.

ppszServerFQDN: A string. If the server does not return an error, the **ppszServerFQDN** prarmeter contains the FQDN of the mailbox server identified by the **szMailboxServerDN** parameter.

Return Values: The server returns 0 for a successful execution. An error results in an HRESULT or other nonzero error code.

Exceptions Thrown: No exceptions are thrown beyond those thrown by the underlying RPC protocol as specified in [MS-RPCE].

The server MUST process the data from the client using the following constraints when receiving this message. The method MUST perform some lookup to determine the FQDN of the server identified by the **szMailboxServerDN** parameter. After considering these constraints, this method SHOULD return one mailbox server name in the **ppszServerFQDN** parameter and 0 as a return value. If any errors occur and the method is not able to return the name of a mailbox server, a failing HRESULT SHOULD be returned.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

4 Protocol Examples

The **RfrGetNewDSA** method is explained in the following example.

The client requests an NSPI server name from the server by calling the **RfrGetNewDSA** method with the **pUserDN** parameter set to the distinguished name (DN) of the client's mailbox.

Typical parameters will look like the following:

```
    // RPC handle returned by RPC binding functions
    hRpc
       0x00010480   handle_t
    ulFlags
       0x00000000   unsigned long
    pUserDN
       "/o=First Organization/ou=Exchange Administrative Group
(FYDIBOHF23SPDLT)/cn=Recipients/cn=user1"      unsigned char *
    ppszUnused
       0x00000000   unsigned char * *
    // memory address which will receive output string
    ppszServer
       0x62348000   unsigned char * *
```

The server responds to the **RfrGetNewDSA** method with a return code of 0 and a valid server name.

Typical parameters will look like the following:

```
    ppszServer   "server1.example.com"   unsigned char * *
```

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations specific to this protocol. General security considerations pertaining to the underlying remote procedure call (RPC)-based transport apply, as described in [MS-RPCE]. This protocol usually requires authentication, but generally does not restrict any caller who is authenticated.

5.2 Index of Security Parameters

Security parameter	Section
Protocol Sequences	2.1

6 Appendix A: Full IDL

For ease of implementation, the following full WSDL is provided, 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 3.1.1.5.1. For example, as noted in [MS-RPCE] section 2.2.4.9, a pointer_default declaration is not required and pointer_default(unique) is assumed.

```
import "ms-dtyp.idl";
[ uuid (1544f5e0-613c-11d1-93df-00c04fd7bd09),
  version(1.0),
  pointer_default(unique)]
interface rfri
{
long RfrGetNewDSA(
 [in]                       handle_t          hRpc,
 [in]                       unsigned long     ulFlags,
 [in, string]               unsigned char *   pUserDN,
 [in,out,unique, string]    unsigned char * * ppszUnused,
 [in,out,unique, string]    unsigned char * * ppszServer);

long RfrGetFQDNFromServerDN(
   [in]                                    handle_t           hRpc,
   [in]                                    unsigned long      ulFlags,
   [in, range(10,1024)]                    unsigned long      cbMailboxServerDN,
   [in, string, size_is(cbMailboxServerDN)] unsigned char   *  szMailboxServerDN,
   [out,ref,string]                        unsigned char   ** ppszServerFQDN);
}
```

7 Appendix B: Product Behavior

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

- Microsoft® Exchange Server 2003
- Microsoft® Exchange Server 2007
- Microsoft® Exchange Server 2010
- Microsoft® Office Outlook® 2003
- Microsoft® Office Outlook® 2007
- Microsoft® Outlook® 2010

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

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

<1> Section 3.1.4.1: The Exchange 2003 and Exchange 2007 implementation of this protocol follow these NSPI server preference rules:

1. Server is up and functioning.
2. Server supports the client's protocol sequence.
3. Server has a writeable copy of the object represented by pUserDN.
4. Server is physically close to the NSPI referral server.

The NSPI servers are compared on these four properties in the order above. If two servers both satisfy or both do not satisfy 1, then 2 is used as a tie-breaker; if two servers both satisfy or both do not satisfy 1 and both satisfy or both don't satisfy 2, then 3 is used as a tie-breaker; and so on. The server that breaks the tie by satisfying a property that the other one does not satisfy is the preferred server. If multiple servers tie after comparing all four properties, those servers are returned in "round robin" order, meaning that each call to RfrGetNewDSA will return the next server in the list of tied servers. In the Exchange 2003 and Exchange 2007 implementation of this protocol, the administrator can configure the protocol to reverse the priorities of properties 3 and 4.

<2> Section 3.1.4.1: Office Outlook 2003 and Office Outlook 2007 can connect to a messaging server with a co-located NSPI server and no NSPI referral server, as well as a messaging server with an NSPI referral server. When first connecting, Office Outlook 2003 and Office Outlook 2007 have not yet determined which type of messaging server they are connecting to, and therefore they will try to connect to the messaging server's co-located NSPI server. On subsequent connections to that server, Office Outlook 2003 and Office Outlook 2007 will use the NSPI referral server. This is one exception to the protocol documentation that states that clients SHOULD always use the NSPI referral server. Clients written to this protocol documentation have no reason to connect to an NSPI server before using this protocol.

<3> Section 3.1.4.2: In Exchange 2003 and Exchange 2007, the `CN=" instance-name "` element is not supported.

8 Change Tracking

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

9 Index

V