

[MS-OXABREF]:

Address Book Name Service Provider Interface (NSPI) Referral Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release version of this technology. This Open Specification is final documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the

documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Preliminary

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.1		Initial Availability.
4/25/2008	0.2		Revised and updated property names and other technical content.
6/27/2008	1.0		Initial Release.
8/6/2008	1.01		Revised and edited technical content.
9/3/2008	1.02		Updated references.
12/3/2008	1.03		Updated IP notice.
4/10/2009	2.0		Updated technical content and applicable product releases.
7/15/2009	3.0	Major	Revised and edited for technical content.
11/4/2009	4.0.0	Major	Updated and revised the technical content.
2/10/2010	4.1.0	Minor	Updated the technical content.
5/5/2010	4.2.0	Minor	Updated the technical content.
8/4/2010	4.3	Minor	Clarified the meaning of the technical content.
11/3/2010	5.0	Major	Significantly changed the technical content.
3/18/2011	6.0	Major	Significantly changed the technical content.
8/5/2011	6.0	No Change	No changes to the meaning, language, or formatting of the technical content.
10/7/2011	6.0	No Change	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	7.0	Major	Significantly changed the technical content.
4/27/2012	7.1	Minor	Clarified the meaning of the technical content.
7/16/2012	7.1	No Change	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	7.2	Minor	Clarified the meaning of the technical content.
2/11/2013	8.0	Major	Significantly changed the technical content.
7/26/2013	8.1	Minor	Clarified the meaning of the technical content.
11/18/2013	8.1	No Change	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	8.1	No Change	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	8.1	No Change	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	8.1	No Change	No changes to the meaning, language, or formatting of the

Date	Revision History	Revision Class	Comments
			technical content.
10/30/2014	8.1	No Change	No changes to the meaning, language, or formatting of the technical content.
3/16/2015	9.0	Major	Significantly changed the technical content.
5/26/2015	9.0	No Change	No changes to the meaning, language, or formatting of the technical content.

Preliminary

Table of Contents

1	Introduction	6
1.1	Glossary	6
1.2	References	7
1.2.1	Normative References	7
1.2.2	Informative References	8
1.3	Overview	8
1.4	Relationship to Other Protocols	9
1.5	Prerequisites/Preconditions	9
1.6	Applicability Statement	9
1.7	Versioning and Capability Negotiation	9
1.8	Vendor-Extensible Fields	10
1.9	Standards Assignments.....	10
2	Messages.....	11
2.1	Transport.....	11
2.2	Common Data Types	11
2.2.1	handle_t.....	11
3	Protocol Details.....	12
3.1	NSPI Referral Server Details.....	12
3.1.1	Abstract Data Model.....	12
3.1.2	Timers	12
3.1.3	Initialization.....	12
3.1.4	Message Processing Events and Sequencing Rules	12
3.1.4.1	RfrGetNewDSA (opnum 0)	13
3.1.4.2	RfrGetFQDNFromServerDN (opnum 1).....	14
3.1.5	Timer Events.....	15
3.1.6	Other Local Events.....	15
4	Protocol Examples.....	16
5	Security.....	17
5.1	Security Considerations for Implementers	17
5.2	Index of Security Parameters	17
6	Appendix A: Full IDL.....	18
7	Appendix B: Product Behavior	19
8	Change Tracking.....	21
9	Index.....	23

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 can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in [\[RFC2119\]](#). Sections 1.5 and 1.9 are also normative but do not contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are specific to this document:

Address Book object: An entity in an address book that contains a set of attributes (1), each attribute with a set of associated values.

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

distinguished name (DN): A name that uniquely identifies an object by using the relative distinguished name (RDN) for the object, and the names of container objects and domains that contain the object. The distinguished name (DN) identifies the object and its location in a tree.

Domain Name System (DNS): A hierarchical, distributed database that contains mappings of domain names (1) to various types of data, such as IP addresses. DNS enables the location of computers and services by user-friendly names, and it also enables the discovery of other information stored in the database.

dynamic endpoint: A network-specific server address that is requested and assigned at run time. For more information, see [\[C706\]](#).

flags: A set of values used to configure or report options or settings.

fully qualified domain name (FQDN): An unambiguous domain name (2) that gives an absolute location in the **Domain Name System's (DNS)** hierarchy tree, as defined in [\[RFC1035\]](#) section 3.1 and [\[RFC2181\]](#) section 11.

Interface Definition Language (IDL): The International Standards Organization (ISO) standard language for specifying the interface for remote procedure calls. For more information, see [\[C706\]](#) section 4.

Kerberos: An authentication (2) system that enables two parties to exchange private information across an otherwise open network by assigning a unique key (called a ticket) to each user that logs on to the network and then embedding these tickets into messages sent by the users. For more information, see [\[MS-KILE\]](#).

mailbox: A message store that contains email, calendar items, and other Message objects for a single recipient.

name service provider interface (NSPI): A method of performing address-book-related operations on Active Directory.

Network Data Representation (NDR): A specification that defines a mapping from **Interface Definition Language (IDL)** data types onto octet streams. **NDR** also refers to the runtime

environment that implements the mapping facilities (for example, data provided to **NDR**). For more information, see [\[MS-RPCE\]](#) and [C706] section 14.

NT LAN Manager (NTLM) Authentication Protocol: A protocol using a challenge-response mechanism for authentication (2) in which clients are able to verify their identities without sending a password to the server. It consists of three messages, commonly referred to as Type 1 (negotiation), Type 2 (challenge) and Type 3 (authentication). For more information, see [\[MS-NLMP\]](#).

opnum: An operation number or numeric identifier that is used to identify a specific **remote procedure call (RPC)** method or a method in an interface. For more information, see [C706] section 12.5.2.12 or [MS-RPCE].

public folder: A Folder object that is stored in a location that is publicly available.

remote procedure call (RPC): A context-dependent term commonly overloaded with three meanings. Note that much of the industry literature concerning RPC technologies uses this term interchangeably for any of the three meanings. Following are the three definitions: (*) The runtime environment providing remote procedure call facilities. The preferred usage for this meaning is "RPC runtime". (*) The pattern of request and response message exchange between two parties (typically, a client and a server). The preferred usage for this meaning is "RPC exchange". (*) A single message from an exchange as defined in the previous definition. The preferred usage for this term is "RPC message". For more information about RPC, see [C706].

RPC protocol sequence: A character string that represents a valid combination of a **remote procedure call (RPC)** protocol, a network layer protocol, and a transport layer protocol, as described in [C706] and [MS-RPCE].

universally unique identifier (UUID): A 128-bit value. UUIDs can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects in cross-process communication such as client and server interfaces, manager entry-point vectors, and **RPC** objects. UUIDs are highly likely to be unique. UUIDs are also known as globally unique identifiers (GUIDs) and these terms are used interchangeably in the Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the UUID. Specifically, the use of this term does not imply or require that the algorithms described in [\[RFC4122\]](#) or [C706] must be used for generating the UUID.

well-known endpoint: A preassigned, network-specific, stable address for a particular client/server instance. For more information, see [C706].

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the [Errata](#).

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.

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, <https://www2.opengroup.org/ogsys/catalog/c706>

[MS-ERREF] Microsoft Corporation, "[Windows Error Codes](#)".

[MS-OXCRPC] Microsoft Corporation, "[Wire Format Protocol](#)".

[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-OXOABK] Microsoft Corporation, "[Address Book Object Protocol](#)".

[MS-OXPROTO] Microsoft Corporation, "[Exchange Server Protocols System Overview](#)".

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 FQDN of the mailbox server, when only the 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.

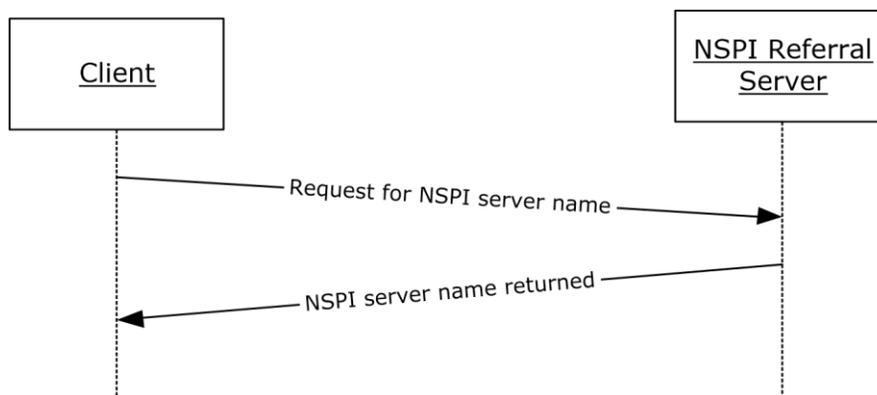


Figure 1 Client retrieving NSPI server name from the NSPI referral server

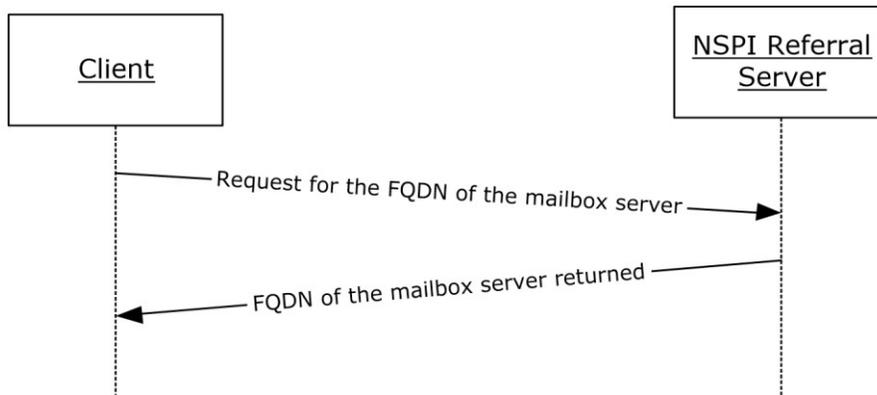


Figure 2 Client retrieving mailbox server name from the NSPI referral server

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 [\[MS-RPCE\]](#).

For conceptual background information and overviews of the relationships and interactions between this and other protocols, see [\[MS-OXPROTO\]](#).

1.5 Prerequisites/Preconditions

None.

1.6 Applicability Statement

This protocol is designed to return the name of a name service provider interface (NSPI) server before the client engages in any NSPI requests. It is also designed to return the FQDN of a mailbox server, as described in [\[RFC1035\]](#), when a client only knows the 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 2.2.4.37) 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, as specified in section [3.1.4.1](#), 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 protocol sequences specified in [\[MS-OXCRPC\]](#) section 2.1.

This protocol uses a well-known endpoint, 6002, for the RPC protocol sequence `ncacn_http`.

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 SHOULD [<1>](#) indicate to the RPC runtime that it is to perform a strict NDR data consistency check at target level 6.0, or it MAY indicate a strict 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) FQDN of the server corresponding to the passed 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 or a server array.

```
//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 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 FQDN of an NSPI server or a server array. 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 server array or a server from the user's mailbox site. <2> 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 connects 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**.

Note that clients 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, the client will not have 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, the client 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.

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) FQDN of the server corresponding to the passed DN.

```
// 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** parameter 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 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

Preliminary

6 Appendix A: Full IDL

For ease of implementation, the following full IDL 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 [\[MS-RPCE\]](#). For example, as noted in [\[MS-RPCE\]](#), 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);
}
```

Preliminary

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 Exchange Server 2013
- Microsoft Exchange Server 2016 Preview
- Microsoft Office Outlook 2003
- Microsoft Office Outlook 2007
- Microsoft Outlook 2010
- Microsoft Outlook 2013
- Microsoft Outlook 2016 Preview

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](#): Windows Vista operating system, Windows Server 2008 operating system, Windows 7 operating system, Windows Server 2008 R2 operating system, and Windows Server 2012 operating system: Specify target level 6.0. Windows 2000 operating system, Windows XP operating system, and Windows Server 2003 operating system: Specify target level 5.0.

[<2> 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.

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

Preliminary

8 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- The removal of a document from the documentation set.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.
- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated**.

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
Z Appendix B: Product Behavior	Added Exchange 2016 and Outlook 2016 to the list of applicable products.	Y	Content update.

Preliminary

9 Index

A

Abstract data model
[server](#) 12
[Applicability](#) 9

C

[Capability negotiation](#) 9
[Change tracking](#) 21
[Common data types](#) 11

D

Data model - abstract
[server](#) 12
Data types
[common - overview](#) 11

E

Events
[local - server](#) 15
[timer - server](#) 15
Examples
[overview](#) 16

F

[Fields - vendor-extensible](#) 10
[Full IDL](#) 18

G

[Glossary](#) 6

I

[IDL](#) 18
[Implementer - security considerations](#) 17
[Index of security parameters](#) 17
[Informative references](#) 8
Initialization
[server](#) 12
Interfaces - server
[nspi referral](#) 12
[Introduction](#) 6

L

Local events
[server](#) 15

M

Message processing
[server](#) 12
Messages
[common data types](#) 11

[transport](#) 11

Methods

[RfrGetFQDNFromServerDN \(opnum 1\)](#) 14
[RfrGetNewDSA \(opnum 0\)](#) 13

N

[Normative references](#) 7
[nspi referral interface](#) 12

O

[Overview \(synopsis\)](#) 8

P

[Parameters - security index](#) 17
[Preconditions](#) 9
[Prerequisites](#) 9
[Product behavior](#) 19
Protocol Details
[overview](#) 12

R

[References](#) 7
[informative](#) 8
[normative](#) 7
[Relationship to other protocols](#) 9
[RfrGetFQDNFromServerDN \(opnum 1\) method](#) 14
[RfrGetNewDSA \(opnum 0\) method](#) 13

S

Security
[implementer considerations](#) 17
[parameter index](#) 17
Sequencing rules
[server](#) 12
Server
[abstract data model](#) 12
[initialization](#) 12
[local events](#) 15
[message processing](#) 12
[nspi referral interface](#) 12
[overview](#) 12
[RfrGetFQDNFromServerDN \(opnum 1\) method](#) 14
[RfrGetNewDSA \(opnum 0\) method](#) 13
[sequencing rules](#) 12
[timer events](#) 15
[timers](#) 12
[Standards assignments](#) 10

T

Timer events
[server](#) 15
Timers
[server](#) 12

[Tracking changes](#) 21
[Transport](#) 11

v

[Vendor-extensible fields](#) 10
[Versioning](#) 9

Preliminary