

[MS-SNID-Diff]:

Server Network Information Discovery Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation (“this documentation”) for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **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 can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](#) or the [Microsoft Community Promise](#). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs.** To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](#).
- **Trademarks.** The names of companies and products contained in this documentation might 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](http://www.microsoft.com/trademarks).
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are 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 as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation 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. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Revision Summary

Date	Revision History	Revision Class	Comments
8/8/2013	1.0	New	Released new document.
11/14/2013	2.0	Major	Updated and revised the technical content.
2/13/2014	2.0	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	2.0	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	2.0	None	No changes to the meaning, language, or formatting of the technical content.
7/14/2016	2.0	None	No changes to the meaning, language, or formatting of the technical content.
<u>6/1/2017</u>	<u>2.0</u>	<u>None</u>	<u>No changes to the meaning, language, or formatting of the technical content.</u>

Table of Contents

1	Introduction	5
1.1	Glossary	5
1.2	References	5
1.2.1	Normative References	5
1.2.2	Informative References	5
1.3	Overview	6
1.4	Relationship to Other Protocols	6
1.5	Prerequisites/Preconditions	6
1.6	Applicability Statement	6
1.7	Versioning and Capability Negotiation	6
1.8	Vendor-Extensible Fields	6
1.9	Standards Assignments	6
2	Messages	7
2.1	Transport	7
2.2	Message Syntax	7
2.2.1	Enumerations	7
2.2.1.1	Id Enumeration	7
2.2.2	Structures	7
2.2.2.1	Network Information Discovery Request	7
2.2.2.2	SOCKADDR_STORAGE	7
2.2.2.2.1	SOCKADDR_IN	8
2.2.2.2.2	SOCKADDR_IN6	9
2.2.2.3	Network Information Discovery Response	9
2.2.3	Namespaces	10
2.2.4	Messages	10
2.3	Directory Service Schema Elements	10
3	Protocol Details	11
3.1	Client Details	11
3.1.1	Abstract Data Model	11
3.1.2	Timers	11
3.1.3	Initialization	11
3.1.4	Higher-Layer Triggered Events	11
3.1.5	Message Processing Events and Sequencing Rules	11
3.1.6	Timer Events	11
3.1.7	Other Local Events	11
3.2	Server Details	11
3.2.1	Abstract Data Model	11
3.2.2	Timers	11
3.2.3	Initialization	11
3.2.4	Higher-Layer Triggered Events	12
3.2.5	Message Processing Events and Sequencing Rules	12
3.2.6	Timer Events	12
3.2.7	Other Local Events	12
4	Protocol Examples	13
5	Security	14
5.1	Security Considerations for Implementers	14
5.2	Index of Security Parameters	14
6	Appendix A: Product Behavior	15
7	Change Tracking	16
8	Index	17

1 Introduction

The Server Network Information Discovery Protocol enables protocol clients to discover protocol servers within a UDP broadcast boundary and [to](#) get server's networking configuration information.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

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.

Internet Protocol version 4 (IPv4): An Internet protocol that has 32-bit source and destination addresses. IPv4 is the predecessor of IPv6.

Internet Protocol version 6 (IPv6): A revised version of the Internet Protocol (IP) designed to address growth on the Internet. Improvements include a 128-bit IP address size, expanded routing capabilities, and support for authentication ~~(2)~~ and privacy.

NetBIOS name: A 16-byte address that is used to identify a NetBIOS resource on the network. For more information, see [RFC1001] and [RFC1002].

User Datagram Protocol (UDP): The connectionless protocol within TCP/IP that corresponds to the transport layer in the ISO/OSI reference model.

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.

[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>

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980, <http://www.ietf/rfc-editor.org/rfc/rfc768.txt>

1.2.2 Informative References

None.

1.3 Overview

The Server Network Information Discovery Protocol defines a pair of request and response messages by which a protocol client can locate protocol servers within the broadcast/multicast scope ~~and. The protocol client can~~ get network information ~~(from the servers,~~ such as NetBIOS name, Internet Protocol version 4 (IPv4), and Internet Protocol version 6 (IPv6) addresses) ~~of the servers.~~

1.4 Relationship to Other Protocols

The protocol relies on User Datagram Protocol (UDP) as specified in [~~RFC-768~~RFC768].

1.5 Prerequisites/Preconditions

The protocol server allows the incoming UDP package from port 8912 in the firewall.

1.6 Applicability Statement

The protocol server and client are connected to the same subnet with IPv4 broadcast support or IPv4 link-local scope multicast support.

1.7 Versioning and Capability Negotiation

The protocol version specifies the interoperability capability of the protocol on different Windows operating systems. See section 6 for detailed product behaviors of protocol versions and corresponding supported operating system. Protocol servers and clients with different protocol versions are not able to negotiate with each other.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

Parameter	Value	Reference
UDP port	8912	http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml

2 Messages

2.1 Transport

The protocol transports messages over UDP as specified in [RFC768]. A protocol client MUST send one UDP package to the broadcast (IPv4) address and multicast (IPv6 link local scope all nodes) address using UDP port 8912. Protocol servers MUST reply to the protocol client with a UDP package containing its NetBIOS name and DNS configuration of the protocol server as specified later in this section.

2.2 Message Syntax

2.2.1 Enumerations

2.2.1.1 Id Enumeration

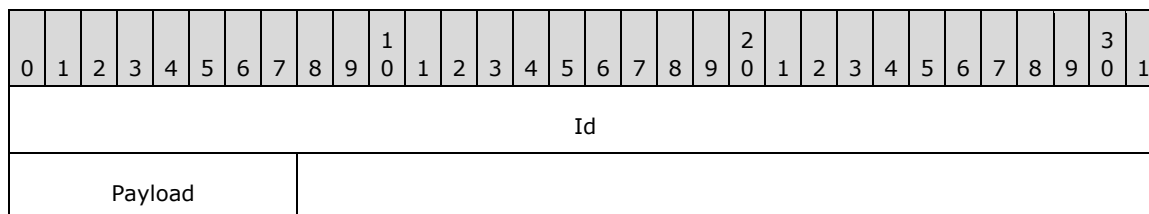
The **Id** enumeration is used to distinguish the Network Information Discovery request and Network Information Discovery response package types.

Field/Value	Description
RequestIdentifier 0x00000000	Indicates that the package is a Network Information Discovery request from the protocol client to the protocol server.
ResponseIdentifier 0xFFFFFFFF	Indicates that the package is a Network Information Discovery response from the protocol server to the protocol client.

2.2.2 Structures

2.2.2.1 Network Information Discovery Request

The package from protocol clients MUST set the **Id** field to RequestIdentifier and SHOULD include a single byte payload of any value.



Id (4 bytes): A message identifier to specify the package type of Network Information Discovery Request. The value MUST be RequestIdentifier, ~~which is~~ defined in 2.2.1.1.

Payload (1 byte): A package payload that contains the value.

2.2.2.2 SOCKADDR_STORAGE

SOCKADDR_STORAGE is a 128-byte structure that is formatted as follows: ±

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Family																Buffer (variable)															
...																															
Reserved (variable)																															
...																															

Family (2 bytes): The address family of the socket. This field MUST contain one of the following values:

Value	Meaning
InterNetwork 0x0002	When set, this indicates an IPv4 address in the socket.
InterNetworkV6 0x0017	When set, this indicates an IPv6 address in the socket.

Buffer (variable): A variable-length buffer that contains the socket address information. If the value of the **Family** field is 0x0002, this field MUST be interpreted as SOCKADDR_IN (section 2.2.2.2.1). Otherwise, if the value of the **Family** field is 0x0017, this field MUST be interpreted as SOCKADDR_IN6 (section 2.2.2.2.2).

Reserved (variable): The remaining bytes within the size of the SOCKADDR_STORAGE structure (128 bytes) MUST NOT be used and MUST be reserved. The server SHOULD set this to zero, and the client MUST ignore it on receipt.

2.2.2.2.1 SOCKADDR_IN

SOCKADDR_IN is a 14-byte structure formatted as follows. All fields in this structure are in network byte order.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Port														IPv4Address																	
...														Reserved																	
...																															
...																															

Port (2 bytes): This field MUST NOT be used and MUST be reserved. The server SHOULD set this field to zero, and the client MUST ignore it on receipt.

IPv4Address (4 bytes): The IPv4 address.

Reserved (8 bytes): This field MUST NOT be used and MUST be reserved. The server SHOULD set this field to zero, and the client MUST ignore it on receipt.

2.2.2.2.2 SOCKADDR_IN6

SOCKADDR_IN6 is a 26-byte structure formatted as follows. All fields in this structure are in network byte order.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Port										FlowInfo																					
...										IPv6Address (16 bytes)																					
...																															
...																															
...										ScopeId																					
...																															

Port (2 bytes): This field MUST NOT be used and MUST be reserved. The server SHOULD set this field to zero, and the client MUST ignore it on receipt.

FlowInfo (4 bytes): The server SHOULD set this field to zero, and the client MUST ignore it on receipt.

IPv6Address (16 bytes): IPv6 address. ~~ScopeId (4 bytes): The server SHOULD set this field to zero, and the client MUST ignore it on receipt.~~

ScopeId (4 bytes): The server SHOULD set this field to zero, and the client MUST ignore it on receipt.

2.2.2.3 Network Information Discovery Response

The package from protocol servers MUST set all fields with the format described in this section. There is no alignment requirement for fields after the variable SERVER_NAME, but all fields after that variable are presented in a specific structure with fixed size.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Id																															
SERVER_NAME (variable)																															
...																															
VERSION																															
LOWEST_VERSION																															
IPv4_DNS_NUM																															

IPv4_DNS_ADDRESS (variable)
...
IPv6_DNS_NUM
IPv6_DNS_ADDRESS (variable)
...

Id (4 bytes): Message identifier to specify the package type of Network Information Discovery Response. The value MUST be ResponseIdentifier, ~~which is~~ defined in section 2.2.1.1.

SERVER_NAME (variable): The protocol server's NetBIOS name in a null-terminated Unicode string.

VERSION (4 bytes): The current version of the protocol package. The protocol server MUST set this field ~~according~~ to the corresponding protocol version it is using. The value for this field MUST be 256 or 512. If the field is set to 256, all fields starting from **IPv4_DNS_NUM** are ignored by the protocol client.

LOWEST_VERSION (4 bytes): The lowest version of the protocol package that the protocol server supports. The value of this field MUST be 256 or 512.

IPv4_DNS_NUM (4 bytes): The number of elements stored in **IPv4_DNS_ADDRESS**. This field is set to a value greater than or equal to 0x00000000 if the following IP address fields are used. It is set to 0xFFFFFFFF to ignore all following fields in the message.

IPv4_DNS_ADDRESS (variable): A list of IP addresses stored in the SOCKADDR_STORAGE structure, 128 bytes each. The total number MUST be equal to **IPv4_DNS_NUM**.

IPv6_DNS_NUM (4 bytes): The number of elements stored in **IPv6_DNS_ADDRESS**. The value MUST be greater than or equal to 0x00000000.

IPv6_DNS_ADDRESS (variable): A list of IP addresses stored in the SOCKADDR_STORAGE structure, 128 bytes each. The total number MUST be equal to **IPv6_DNS_NUM**.

2.2.3 Namespaces

None.

2.2.4 Messages

Message	Description
Network information discovery request	The User Datagram Protocol (UDP) broadcast packages are sent from protocol clients to discover protocol servers within the broadcast scope.
Network information discovery response	The UDP packages are sent from protocol servers to protocol clients to respond to the discovery package and to carry the protocol server network information.

2.3 Directory Service Schema Elements

None.

3 Protocol Details

3.1 Client Details

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

The protocol client creates a socket on a randomly selected port and sends a Network Information Discovery Request with [appropriate](#) Id (section 2.2.1.1) defined and a payload to the IPv4 broadcast address and IPv6 link-local all nodes multicast address.

The protocol client then listens to the same port waiting for the Network Information Discovery Response.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

The protocol server creates a UDP socket that listens to port 8912 when the protocol is started.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

When the server receives a UDP Network Information Discovery request package from port 8912, it checks the **Id** in the package. If the **Id** is not a correct request Id as defined in section 2.2.1.1, the package is ignored. If the **Id** is the correct request **Id** as defined in section 2.2.1.1, the server finds the NetBIOS name and all DNS addresses on its network adapters and then sets the corresponding fields of the Network Information Discovery response package.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

Network Information Discovery Request

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x00000000																															
0x01																															

Network Information Discovery Response

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xFFFFFFFF																															
s	v	r	n	a	m	e	\0	0x0000																							
0x0100																0x0000															
0x0100																0x00															
0x0004																															
...																															
0x0006																															
...																															

The example response package's **IPv4_DNS_ADDRESS** field contains four elements of structure **SOCKADDR_STORAGE**, and the **IPv6_DNS_ADDRESS** field contains six elements of structure **SOCKADDR_STORAGE**.

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: 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.

- Windows Home Server 2011 server software
- Windows Small Business Server 2011 Essentials
- Windows Server 2012 operating system
- Windows Server 2012 R2 operating system
- Windows Server 2016 operating system

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.

7 Change Tracking

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

8 Index

A

- Abstract data model 11
 - client 11
 - server 11
- Applicability 6

C

- Capability negotiation 6
- Change tracking 16
- Client
 - abstract data model 11
 - higher-layer triggered events 11
 - initialization 11
 - message processing 11
 - other local events 11
 - sequencing rules 11
 - timer events 11
 - timers 11

D

- Data model - abstract
 - client 11
 - server 11
- Directory service schema elements 10

E

- Elements - directory service schema 10
- Examples - overview 13

F

- Fields - vendor-extensible 6

G

- Glossary 5

H

- Higher-layer triggered events 12
 - client 11
 - server 12

I

- Id enumeration 7
- Implementer - security considerations 14
- Index of security parameters 14
- Informative references 5
- Initialization 11
 - client 11
 - server 11
- Introduction 5

L

- Local events 12

M

- Message processing 12
 - client 11
 - server 12
- Messages
 - Messages 10
 - Namespaces 10
 - network information discovery request 10
 - network information discovery response 10
 - transport 7
- Messages message 10

N

- Namespaces 10
- Namespaces message 10
- Network information discovery request 10
- Network Information Discovery Request packet 7
- Network information discovery response 10
- Normative references 5

O

- Other local events
 - client 11
 - server 12
- Overview (synopsis) 6

P

- packet 9
- Parameters - security index 14
- Preconditions 6
- Prerequisites 6
- Product behavior 15
- Protocol example
 - network information discovery request 13
 - network information discovery response 13

R

- References 5
 - informative 5
 - normative 5
- Relationship to other protocols 6

S

- Schema elements - directory service 10
- Security
 - implementer considerations 14
 - parameter index 14
- Sequencing rules
 - client 11
 - server 12
- Server
 - abstract data model 11
 - higher-layer triggered events 12
 - initialization 11
 - message processing 12
 - other local events 12
 - sequencing rules 12

- timer events 12
- timers 11
- SOCKADDR_IN packet 8
- SOCKADDR_IN6 packet 9
- SOCKADDR_STORAGE packet 7
- Standards assignments 6

T

- Timer events 12
 - client 11
 - server 12
- Timers 11
 - client 11
 - server 11
- Tracking changes 16
- Transport 7
- Triggered events - higher-layer
 - client 11
 - server 12

V

- Vendor-extensible fields 6
- Versioning 6