# [MS-DSSP]:

# Directory Services Setup Remote 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.
- **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 |
|---|---|---|---|
| 3/2/2007 | 1.0 | New | Version 1.0 release |
| 4/3/2007 | 1.1 | Minor | Version 1.1 release |
| 5/11/2007 | 1.2 | Minor | Version 1.2 release |
| 6/1/2007 | 1.2.1 | Editorial | Changed language and formatting in the technical content. |
| 7/3/2007 | 1.3 | Minor | Clarified the meaning of the technical content. |
| 8/10/2007 | 1.4 | Minor | Clarified the meaning of the technical content. |
| 9/28/2007 | 1.5 | Minor | Clarified the meaning of the technical content. |
| 10/23/2007 | 2.0 | Major | Converted document to unified format. |
| 1/25/2008 | 2.0.1 | Editorial | Changed language and formatting in the technical content. |
| 3/14/2008 | 2.0.2 | Editorial | Changed language and formatting in the technical content. |
| 6/20/2008 | 2.1 | Minor | Clarified the meaning of the technical content. |
| 7/25/2008 | 2.1.1 | Editorial | Changed language and formatting in the technical content. |
| 8/29/2008 | 2.2 | Minor | Clarified the meaning of the technical content. |
| 10/24/2008 | 2.2.1 | Editorial | Changed language and formatting in the technical content. |
| 12/5/2008 | 2.3 | Minor | Clarified the meaning of the technical content. |
| 1/16/2009 | 2.4 | Minor | Clarified the meaning of the technical content. |
| 2/27/2009 | 2.4.1 | Editorial | Changed language and formatting in the technical content. |
| 4/10/2009 | 2.4.2 | Editorial | Changed language and formatting in the technical content. |
| 5/22/2009 | 2.4.3 | Editorial | Changed language and formatting in the technical content. |
| 7/2/2009 | 2.5 | Minor | Clarified the meaning of the technical content. |
| 8/14/2009 | 2.5.1 | Editorial | Changed language and formatting in the technical content. |
| 9/25/2009 | 2.6 | Minor | Clarified the meaning of the technical content. |
| 11/6/2009 | 3.0 | Major | Updated and revised the technical content. |
| 12/18/2009 | 3.1 | Minor | Clarified the meaning of the technical content. |
| 1/29/2010 | 4.0 | Major | Updated and revised the technical content. |
| 3/12/2010 | 4.1 | Minor | Clarified the meaning of the technical content. |
| 4/23/2010 | 4.2 | Minor | Clarified the meaning of the technical content. |
| 6/4/2010 | 4.3 | Minor | Clarified the meaning of the technical content. |
| 7/16/2010 | 4.3 | None | No changes to the meaning, language, or formatting of the technical content. |
| 8/27/2010 | 4.3 | None | No changes to the meaning, language, or formatting of the |

| Date | Revision History | Revision Class | Comments |
|---|---|---|---|
| | | | technical content. |
| 10/8/2010 | 4.3 | None | No changes to the meaning, language, or formatting of the technical content. |
| 11/19/2010 | 4.3 | None | No changes to the meaning, language, or formatting of the technical content. |
| 1/7/2011 | 4.3 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/11/2011 | 5.0 | Major | Updated and revised the technical content. |
| 3/25/2011 | 6.0 | Major | Updated and revised the technical content. |
| 5/6/2011 | 7.0 | Major | Updated and revised the technical content. |
| 6/17/2011 | 7.1 | Minor | Clarified the meaning of the technical content. |
| 9/23/2011 | 7.1 | None | No changes to the meaning, language, or formatting of the technical content. |
| 12/16/2011 | 8.0 | Major | Updated and revised the technical content. |
| 3/30/2012 | 8.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/12/2012 | 8.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/25/2012 | 9.0 | Major | Updated and revised the technical content. |
| 1/31/2013 | 9.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 8/8/2013 | 10.0 | Major | Updated and revised the technical content. |
| 11/14/2013 | 10.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/13/2014 | 10.1 | Minor | Clarified the meaning of the technical content. |
| 5/15/2014 | 10.1 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/30/2015 | 11.0 | Major | Significantly changed the technical content. |
| 10/16/2015 | 11.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/14/2016 | 11.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/1/2017 | 11.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 9/15/2017 | 12.0 | Major | Significantly changed the technical content. |
| 9/12/2018 | 13.0 | Major | Significantly changed the technical content. |

# Table of Contents

# 1 Introduction

The Directory Services Setup Remote Protocol is a client/server-based **remote procedure call (RPC)** protocol. The protocol exposes an RPC interface that a client can call to obtain **domain**-related computer state and 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:

**Active Directory**: The Windows implementation of a general-purpose **directory service**, which uses LDAP as its primary access protocol. **Active Directory** stores information about a variety of objects in the network such as user accounts, computer accounts, groups, and all related credential information used by Kerberos [MS-KILE]. **Active Directory** is either deployed as Active Directory Domain Services (AD DS) or Active Directory Lightweight Directory Services (AD LDS), which are both described in [MS-ADOD]: Active Directory Protocols Overview.

**Active Directory domain**: A **domain** hosted on **Active Directory**. For more information, see [MS-ADTS].

**backup domain controller (BDC)**: A **domain controller (DC)** that receives a copy of the **domain** directory database from the **primary domain controller (PDC)**. This copy is synchronized periodically and automatically with the **primary domain controller (PDC)**. BDCs also authenticate user logons and can be promoted to function as the **PDC**. There is only one **PDC** or **PDC** emulator in a **domain**, and the rest are **backup domain controllers**.

**directory**: The database that stores information about objects such as users, groups, computers, printers, and the **directory service** that makes this information available to users and applications.

**directory service (DS)**: A service that stores and organizes information about a computer network's users and network shares, and that allows network administrators to manage users' access to the shares. See also **Active Directory**.

**domain**: A set of users and computers sharing a common namespace and management infrastructure. At least one computer member of the set must act as a **domain controller (DC)** and host a member list that identifies all members of the domain, as well as optionally hosting the **Active Directory** service. The domain controller provides authentication of members, creating a unit of trust for its members. Each domain has an identifier that is shared among its members. For more information, see [MS-AUTHSOD] section 1.1.1.5 and [MS-ADTS].

**domain controller (DC)**: The service, running on a server, that implements **Active Directory**, or the server hosting this service. The service hosts the data store for objects and interoperates with other **DCs** to ensure that a local change to an object replicates correctly across all **DCs**. When **Active Directory** is operating as Active Directory Domain Services (AD DS), the **DC** contains full NC replicas of the configuration naming context (config NC), schema naming context (schema NC), and one of the domain NCs in its **forest**. If the AD DS **DC** is a global catalog server (GC server), it contains partial NC replicas of the remaining domain NCs in its **forest**. For more information, see [MS-AUTHSOD] section 1.1.1.5.2 and [MS-ADTS]. When **Active Directory** is operating as Active Directory Lightweight Directory Services (AD LDS), several AD LDS **DCs** can run on one server. When **Active Directory** is operating as AD DS, only one AD DS **DC** can run on one server. However, several AD LDS **DCs** can coexist with one AD DS **DC** on one server. The AD LDS **DC** contains full NC replicas of the config NC and the schema NC in its **forest**. The domain controller is the server side of Authentication Protocol Domain Support [MS-APDS].

**domain membership role**: Quantifies the relationship between a computer and a domain. A computer can act in one of three roles: (1) Joined -- linked to a domain for purposes of policy and security; (2) Stand-alone -- not associated with any domain; or (3) **Domain controller** -- linked to a domain and hosting that domain.

**domain membership role change**: It is possible to change the **domain membership role** of a computer. A stand-alone computer can become a domain-joined computer and vice versa. A computer that is not a **domain controller** can become a **domain controller**, and vice versa.

**endpoint**: A client that is on a network and is requesting access to a network access server (NAS).

**forest**: One or more **domains** that share a common schema and trust each other transitively. An organization can have multiple **forests**. A **forest** establishes the security and administrative boundary for all the objects that reside within the **domains** that belong to the **forest**. In contrast, a **domain** establishes the administrative boundary for managing objects, such as users, groups, and computers. In addition, each **domain** has individual security policies and trust relationships with other **domains**.

**fully qualified domain name (FQDN)**: An unambiguous domain name 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.

**globally unique identifier (GUID)**: A term used interchangeably with **universally unique identifier (UUID)** in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. 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 **GUID**. See also **universally unique identifier (UUID)**.

**legacy domain**: A domain in which all the **domain controllers** are **legacy domain controllers**.

**legacy domain controller**: A **domain controller** that supports the Security Account Manager Remote Protocol [MS-SAMR], but not the **Active Directory** protocols specified in [MS-ADTS] and [MS-DRSR].

**Microsoft Interface Definition Language (MIDL)**: The Microsoft implementation and extension of the OSF-DCE Interface Definition Language (IDL). **MIDL** can also mean the Interface Definition Language (IDL) compiler provided by Microsoft. For more information, see [MS-RPCE].

**mixed mode**: A state of an **Active Directory domain** that supports **domain controllers (DCs)** running Windows NT Server 4.0 operating system. Mixed mode does not allow organizations to take advantage of new **Active Directory** features such as universal groups, nested group membership, and interdomain group membership. See also **native mode**.

**native mode**: A state of an **Active Directory** domain in which all current and future **domain controllers (DCs)** use AD style domains. **Native mode** allows organizations to take advantage of the new **Active Directory** features such as universal groups, nested group membership, and interdomain group membership.

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

**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.

**operating system upgrade**: The action of replacing the existing operating system on a computer with a later version of the operating system while maintaining the original configuration and data of that computer.

**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].

**primary domain controller (PDC)**: A **domain controller (DC)** designated to track changes made to the accounts of all computers on a **domain**. It is the only computer to receive these changes directly, and is specialized so as to ensure consistency and to eliminate the potential for conflicting entries in the **Active Directory** database. A **domain** has only one **PDC**.

**primary domain controller (PDC) role owner**: The **domain controller (DC)** that hosts the **primary domain controller** emulator FSMO role for a given domain naming context (NC).

**read-only domain controller (RODC)**: A **domain controller (DC)** that does not accept originating updates. Additionally, an **RODC** does not perform outbound replication. An RODC cannot be the primary domain controller (PDC) for its domain.

**remote procedure call (RPC)**: A communication protocol used primarily between client and server. The term has three definitions that are often used interchangeably: a runtime environment providing for communication facilities between computers (the RPC runtime); a set of request-and-response message exchanges between computers (the RPC exchange); and the single message from an RPC exchange (the RPC message). For more information, see [C706].

**RPC transport**: The underlying network services used by the remote procedure call (RPC) runtime for communications between network nodes. For more information, see [C706] section 2.

**Server Message Block (SMB)**: A protocol that is used to request file and print services from server systems over a network. The SMB protocol extends the CIFS protocol with additional security, file, and disk management support. For more information, see [CIFS] and [MS-SMB].

**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-ADTS] Microsoft Corporation, "Active Directory Technical Specification".

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

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

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

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

[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

None.

## 1.3 Overview

This protocol provides a **remote procedure call (RPC)** interface for querying domain-related computer state and configuration data. The client end of the Directory Services Setup Remote Protocol is an application that issues method calls on the RPC interface. The server end of the Directory Services Setup Remote Protocol obtains and replies to the client with the requested data about the computer on which the server is running. If the client connects to and requests information about a **domain controller (DC)** for the **directory service**, this data includes the status of any pending promotion or demotion of that DC.

## 1.4 Relationship to Other Protocols

The Directory Services Setup Remote Protocol is dependent upon Microsoft **remote procedure call (RPC)** (Remote Procedure Call Protocol Extensions, as specified in [MS-RPCE]), which is used to communicate between computers on a network.

This protocol depends on the **Server Message Block (SMB)** Protocol, as specified in [MS-SMB], and TCP/IP protocols for sending messages on the wire.

## 1.5 Prerequisites/Preconditions

This protocol is a **remote procedure call (RPC)**-based protocol and therefore has the prerequisites, as specified in [MS-RPCE], common to all RPC interfaces.

Security considerations for RPC usage are specified in section 5.1.

## 1.6 Applicability Statement

This protocol can be used to perform the following functions:

- Obtain the configuration information of the domain to which a computer is joined. The information includes the domain name and domain **globally unique identifier (GUID)**. This protocol can be used to query a **DC** to determine if it is a **primary domain controller (PDC)** (or **primary domain controller (PDC) role owner**) or a **read-only domain controller**.
- Query the progress of the promotion or demotion of a DC.

- Retrieve the upgrade status of a DC. This information is only applicable for the upgrade of a **legacy domain controller** to a version of Windows that is able to host **Active Directory**.
- Retrieve the **domain membership role** type for the computer.

## 1.7 Versioning and Capability Negotiation

**Supported Transports:** This protocol uses only **RPCs**. The protocol supports the **Server Message Block (SMB)** transport. For more information, see section 2.1.

- **Protocol Version:** This protocol interface has a single version number of 0.0. An RPC client determines whether a method is supported by attempting to call the method; if the method is not supported, the RPC server will return an "Opnum out of range" error<1> as specified in [C706] and [MS-RPCE].
- **Security and Authentication Methods:** Authentication and security are provided as specified in [MS-SMB] and [MS-RPCE]. Anonymous access can be allowed for some operations, as specified in DsRolerGetPrimaryDomainInformation (Opnum 0) (section 3.2.5.1).

## 1.8 Vendor-Extensible Fields

This protocol does not define any vendor-extensible fields within the protocol itself.

## 1.9 Standards Assignments

| Parameter | Value | Reference |
|---|---|---|
| Named pipe | \PIPE\lsarpc | Section 2.1 |
| **RPC** Interface **UUID** for Directory Services Setup Remote Protocol | 3919286a-b10c-11d0-9ba8-00c04fd92ef5 | Section 2.1 |

No public standard assignments have been received for this protocol. All values used in these extensions are in private ranges specified in section 2.1.

# 2 Messages

## 2.1 Transport

This protocol MUST use the following **remote procedure call (RPC)** protocol sequence: RPC over **SMB** (ncacn_np), as specified in [MS-RPCE].

This protocol uses the following **well-known endpoints**. These **endpoints** are pipe names for RPC over SMB, as specified in [MS-RPCE]:

- \PIPE\lsarpc

A server MUST listen on RPC over the above-named pipe. A client MUST only attempt to connect to this protocol via RPC over the above-named pipe.<2>

For authentication and authorization services, both the requestor and responder of this protocol MUST use the SMB transport to communicate the identity of the requestor, as specified in [MS-SMB] section 3.2.4.2.4.

The requestor MUST NOT use the RPC-provided security-support-provider mechanisms (for authentication, authorization, confidentiality, or tamper-resistance services).

This protocol MUST use this **universally unique identifier (UUID)** interface (3919286a-b10c-11d0-9ba8-00c04fd92ef5). The interface version number is 0.0.

## 2.2 Common Data Types

In addition to **RPC** base types, the sections that follow use the definition of **GUID** as specified in [MS-DTYP] Appendix A.

Additional data types that follow are defined in the **Microsoft Interface Definition Language (MIDL)** (as specified in section 6) for this RPC interface.

### 2.2.1 DSROLER_PRIMARY_DOMAIN_INFO_BASIC

The DSROLER_PRIMARY_DOMAIN_INFO_BASIC structure contains basic information, including the role of the computer, domain name, and **GUID** of the domain.

```
typedef struct _DSROLER_PRIMARY_DOMAIN_INFO_BASIC {
  DSROLE_MACHINE_ROLE MachineRole;
  unsigned __int32 Flags;
  [unique, string] wchar_t* DomainNameFlat;
  [unique, string] wchar_t* DomainNameDns;
  [unique, string] wchar_t* DomainForestName;
  GUID DomainGuid;
} DSROLER_PRIMARY_DOMAIN_INFO_BASIC,
 *PDSROLER_PRIMARY_DOMAIN_INFO_BASIC;
```

**MachineRole:** The current role of the computer, expressed as a DSROLE_MACHINE_ROLE data type.

**Flags:** The value that indicates the state of the **directory service** and validity of the information contained in the **DomainGuid** member. The value of this parameter MUST be zero or a combination of one or more individual flags in the following table. The combination is the result of a bitwise OR of the flags that apply to the computer for which information is being retrieved. All undefined bits MUST be 0.

| Value | Meaning |
|---|---|
| DSROLE_PRIMARY_DS_RUNNING<br>0x00000001 | The directory service is running on this computer. If this flag is not set, the directory service is not running on this computer. |
| DSROLE_PRIMARY_DS_MIXED_MODE<br>0x00000002 | The directory service is running in **mixed mode**. This flag is valid only if the DSROLE_PRIMARY_DS_RUNNING flag is set and the DSROLE_PRIMARY_DS_READONLY flag is not set. |
| DSROLE_PRIMARY_DS_READONLY<br>0x00000008 | The computer holds a read-only copy of the **directory**. This flag is valid only if the DSROLE_PRIMARY_DS_RUNNING flag is set and the DSROLE_PRIMARY_DS_MIXED_MODE flag is not set. |
| DSROLE_PRIMARY_DOMAIN_GUID_PRESENT<br>0x01000000 | The **DomainGuid** member contains a valid domain GUID. If this bit is not set, the value in DomainGuid member is undefined. |

**DomainNameFlat:** The **NetBIOS name** of the domain or non-domain workgroup to which the computer belongs.

**DomainNameDns:** The domain name of the computer. This member MUST be NULL if the **MachineRole** member is **DsRole_RoleStandaloneWorkstation** or **DsRole_RoleStandaloneServer** and MUST NOT be NULL otherwise.

**DomainForestName:** The name of the **forest** to which the computer belongs. This member MUST be NULL, if the computer is a stand-alone workstation or server.

**DomainGuid:** The **UUID** of the domain to which the computer belongs. The value of this member is valid only if the DSROLE_PRIMARY_DOMAIN_GUID_PRESENT flag is set.

### 2.2.2 DSROLE_MACHINE_ROLE

The DSROLE_MACHINE_ROLE enumeration specifies the current role of the computer.

```
typedef  enum  DSROLE MACHINE ROLE
{
  DsRole_RoleStandaloneWorkstation,
  DsRole_RoleMemberWorkstation,
  DsRole RoleStandaloneServer,
  DsRole_RoleMemberServer,
  DsRole RoleBackupDomainController,
  DsRole_RolePrimaryDomainController
} DSROLE_MACHINE_ROLE;
```

**DsRole_RoleStandaloneWorkstation:** The computer is a stand-alone workstation.

**DsRole_RoleMemberWorkstation:** The computer is a workstation that is joined to a domain.

**DsRole_RoleStandaloneServer:** The computer is a stand-alone server.

**DsRole_RoleMemberServer:** The computer is a server that is joined to a domain.

**DsRole_RoleBackupDomainController:** The computer is a server that is a **backup domain controller** or a **read-only domain controller**.<3>

**DsRole_RolePrimaryDomainController:** The computer is a server that is the **primary domain controller** emulator.

### 2.2.3 DSROLE_OPERATION_STATE_INFO

The DSROLE_OPERATION_STATE_INFO structure contains the status of a pending **domain controller (DC) domain membership role change** operation, if any, for the computer.

```
typedef struct  DSROLE_OPERATION_STATE_INFO {
  DSROLE_OPERATION_STATE OperationState;
} DSROLE_OPERATION_STATE_INFO,
 *PDSROLE_OPERATION_STATE_INFO;
```

**OperationState:** The domain membership role change status of the computer, as specified by a DSROLE_OPERATION_STATE enumeration.

### 2.2.4 DSROLE_OPERATION_STATE

The DSROLE_OPERATION_STATE enumeration specifies values that determine whether a **DC** promotion or demotion operation is currently being performed on a computer.<4>

```
typedef  enum _DSROLE_OPERATION_STATE
{
  DsRoleOperationIdle = 0,
  DsRoleOperationActive,
  DsRoleOperationNeedReboot
} DSROLE_OPERATION_STATE;
```

**DsRoleOperationIdle:** No promotion or demotion operation is currently being performed on the computer.

**DsRoleOperationActive:** A promotion or demotion operation is in progress.

**DsRoleOperationNeedReboot:** A promotion or demotion operation has been performed. The computer MUST be restarted to function in the new role.

### 2.2.5 DSROLE_UPGRADE_STATUS_INFO

The DSROLE_UPGRADE_STATUS_INFO structure contains information about the status of a pending **operating system upgrade**, if any, for the computer. This structure is intended to store only the status of an operating system upgrade of a **legacy domain controller**.

```
typedef struct _DSROLE_UPGRADE_STATUS_INFO {
  unsigned   int32 OperationState;
  DSROLE_SERVER_STATE PreviousServerState;
} DSROLE_UPGRADE_STATUS_INFO,
 *PDSROLE_UPGRADE_STATUS_INFO;
```

**OperationState:** The current status of the upgrade. Valid values are shown in the following table.<5>

| Value | Meaning |
|---|---|
| 0x00000000 | No upgrade is currently in progress. |
| DSROLE_UPGRADE_IN_PROGRESS<br>0x00000004 | An upgrade is currently in progress. |

**PreviousServerState:** The role of the computer prior to the upgrade. The value of this member is valid only if an upgrade is in progress (that is, if the **OperationState** member is set to DSROLE_UPGRADE_IN_PROGRESS).

### 2.2.6 DSROLE_SERVER_STATE

The DSROLE_SERVER_STATE enumeration specifies the role of the computer prior to the upgrade.

```
typedef  enum  DSROLE SERVER STATE
{
  DsRoleServerUnknown = 0,
  DsRoleServerPrimary,
  DsRoleServerBackup
} DSROLE_SERVER_STATE,
 *PDSROLE_SERVER_STATE;
```

**DsRoleServerUnknown:** The previous role of the computer is unknown.

**DsRoleServerPrimary:** The previous role of the computer was **primary domain controller** in a **legacy domain**.

**DsRoleServerBackup:** The previous role of the computer was **backup domain controller** in a legacy domain.

### 2.2.7 DSROLE_PRIMARY_DOMAIN_INFO_LEVEL

The DSROLE_PRIMARY_DOMAIN_INFO_LEVEL enumeration defines the information level that the client requests.

```
typedef  enum _DSROLE_PRIMARY_DOMAIN_INFO_LEVEL
{
  DsRolePrimaryDomainInfoBasic = 1,
  DsRoleUpgradeStatus,
  DsRoleOperationState
} DSROLE_PRIMARY_DOMAIN_INFO_LEVEL;
```

**DsRolePrimaryDomainInfoBasic:** Request for information about the domain to which the computer belongs.

**DsRoleUpgradeStatus:** Request for computer **operating system upgrade** status.

**DsRoleOperationState:** Request for computer operation state.

### 2.2.8 DSROLER_PRIMARY_DOMAIN_INFORMATION

The DSROLER_PRIMARY_DOMAIN_INFORMATION union contains one of three types of information about a computer.

```
typedef
[switch_type(DSROLE_PRIMARY_DOMAIN_INFO_LEVEL)]
union  DSROLER PRIMARY DOMAIN INFORMATION {
  [case(DsRolePrimaryDomainInfoBasic)]
    DSROLER_PRIMARY_DOMAIN_INFO_BASIC DomainInfoBasic;
  [case(DsRoleUpgradeStatus)]
    DSROLE UPGRADE STATUS INFO UpgradStatusInfo;
  [case(DsRoleOperationState)]
    DSROLE OPERATION STATE INFO OperationStateInfo;
} DSROLER_PRIMARY_DOMAIN_INFORMATION,
```

```
*PDSROLER_PRIMARY_DOMAIN_INFORMATION;
```

**DomainInfoBasic:** Basic information about a computer. For more information, see DSROLER_PRIMARY_DOMAIN_INFO_BASIC (section 2.2.1).

**UpgradStatusInfo:** Information about the upgrade of the computer. For more information, see DSROLE_UPGRADE_STATUS_INFO (section 2.2.5).

**OperationStateInfo:** **Domain membership role change** status of the computer. For more information, see DSROLE_OPERATION_STATE_INFO (section 2.2.3).

## 2.3 Directory Service Schema Elements

None.

# 3 Protocol Details

## 3.1 Client 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.

### 3.1.1 Abstract Data Model

No abstract data model is used.

### 3.1.2 Timers

No protocol timers are required other than those internal ones used in **RPC** to implement resiliency to network outages, as specified in [MS-RPCE].

### 3.1.3 Initialization

No initialization is performed by the client side of the Directory Services Setup Remote Protocol. The **RPC** association (or binding)<6> to the server RPC needed to call the methods of this protocol is performed by the client application. The client side of the Directory Services Setup Remote Protocol simply uses the association established by the client application to call the RPC methods. The details of RPC binding can be found in [MS-RPCE] section 3. The client application MUST create a separate association for each method invocation.

### 3.1.4 Higher-Layer Triggered Events

No higher-layer triggered events are used.

### 3.1.5 Message Processing Events and Sequencing Rules

No special message processing is required on the client beyond the processing required in the underlying **RPC** protocol.

### 3.1.6 Timer Events

No protocol timer events are required on the client other than the events maintained in the underlying **RPC transport**.

### 3.1.7 Other Local Events

No additional local events are used on the client other than the events maintained in the underlying **RPC transport**.

## 3.2 Server Details

### 3.2.1 Abstract Data Model

The following information is maintained by the server to respond to client queries.

The computer maintains abstract variables that contain the identity of the **directory service** domain and forest to which it belongs, if any. The variables are as follows:

**NetBIOSDomainName:** The name of the domain or nondomain workgroup, as known by **NetBIOS name**, to which the computer belongs.

**DNSDomainName:** The **fully qualified domain name (FQDN)** of the domain to which the computer belongs. This abstract element has value only for computers that are joined to a domain; otherwise, it is NULL.

**ForestName:** The FQDN of the forest to which the computer belongs. This variable has value only for computers that are joined to a domain; otherwise, it is NULL.

**DomainGUID:** The **UUID**, as specified in [MS-DTYP], that identifies the domain to which the computer belongs. This variable has type **GUID**, as specified in [MS-DTYP],<7> and has value only for computers that are joined to a directory service domain; otherwise, the value is NULL.

The computer maintains information about its role and status in the domain, as follows:

**ComputerRole (Public):** An abstract variable of type DSROLE_MACHINE_ROLE that describes the current **domain membership role** of the machine.

**ComputerOperationState:** The status of the current ComputerRole change operation. The type of this variable is DSROLE_OPERATION_STATE enumeration.

**ComputerUpgrade:** A Boolean abstract variable that is TRUE only when an upgrade event (as specified in section 3.2.4.3) is in progress.

**PreviousServerState:** The type of this variable is DSROLE_SERVER_STATE enumeration. When ComputerUpgrade is TRUE, it contains the security role that the **domain controller (DC)** will have after the upgrade event (as specified in section 3.2.4.3) is complete. When ComputerUpgrade is FALSE, it contains DsRoleServerUnknown.

### 3.2.2 Timers

No protocol timer events are required on the server other than the timers required in the underlying **RPC transport**, as specified in [MS-RPCE].

### 3.2.3 Initialization

The server MUST listen on the **well-known endpoint** that is defined for this **RPC** interface. For more information, see section 2.1.<8>

ComputerUpgrade is initialized to FALSE.

PreviousServerState is initialized to DsRoleServerUnknown.

ComputerOperationState is initialized to DsRoleOperationIdle.

ComputerRole is set only during initialization. It is initialized as follows:

- If the server meets the requirements of a domain controller as described in [MS-ADTS] section 6.1.2.1, then
  - If the server is hosting the PdcEmulationMasterRole ([MS-ADTS] section 3.1.1.1.11), ComputerRole is set to DsRole_RolePrimaryDomainController, else ComputerRole is set to DsRole_RoleBackupDomainController. The server determines if it is hosting the PdcEmulationMasterRole by invoking the **IsEffectiveRoleOwner** function with the *roleObject* parameter set to RoleObject(Default NC, PdcEmulationMasterRole) (see [MS-ADTS] section 3.1.1.5.1.8).

- Else
  - If DNSDomainName is not NULL, then ComputerRole is set to DsRole_RoleMemberServer, else ComputerRole is set to DsRoleStandaloneServer.

### 3.2.4 Higher-Layer Triggered Events

#### 3.2.4.1 Promotion

Promotion is the act of configuring a server operating system to be a domain controller. At the beginning of promotion, ComputerOperationState MUST be set to DsRoleOperationActive. At the end of promotion, ComputerOperationState MUST be set to DsRoleOperationNeedReboot. Finally, all protocols on the server MUST be reinitialized to complete promotion.<9> The appropriate states of ComputerOperationState and ComputerRole are set during initialization according to section 3.2.3, regardless of the state of a promotion.

The operation or set of operations that constitute promotion (that configure a server operating system to be a domain controller) are server-to-server operations and are not included in this document and are not required for interoperation with clients. The required configuration for successful promotion is the abstract state required of a domain controller's existence as described in [MS-ADTS] section 6.1.2.1.

#### 3.2.4.2 Demotion

Demotion is the act of configuring a domain controller to no longer be a domain controller. At the beginning of demotion, ComputerOperationState MUST be set to DsRoleOperationActive. At the end of demotion, ComputerOperationState MUST be set to DsRoleOperationNeedReboot. Finally, all protocols on the server MUST be reinitialized to complete demotion.<10> The appropriate states of ComputerOperationState and ComputerRole are set during initialization according to section 3.2.3, regardless of the state of a demotion.

The operation or set of operations that constitute demotion (that configure a domain controller to no longer be a domain controller) are server-to-server operations and are not included in this document, and are not required for interoperation with clients.

#### 3.2.4.3 Upgrade

Upgrade is the act of promotion using values suggested from a previously existing source.<11> No upgrade-specific constraints are applied to these values; for example, the **NetBIOS name** of the new domain is not required to match that of a legacy domain. An implementation can choose any specific values as part of promotion as long as the result satisfies the abstract state required of a domain controller's existence as described in [MS-ADTS] section 6.1.2.1.

When the upgrade event begins:

- A promotion event MUST be triggered.
- ComputerUpgrade MUST be set to TRUE.
- PreviousServerState MUST be set to DsRoleServerPrimary if it is promoting the first domain controller in the domain; otherwise, PreviousServerState MUST be set to DsRoleServerBackup. Note that if this event is promoting the first domain controller in the domain, after promotion ComputerRole will be set to DsRole_RolePrimaryDomainController; otherwise, after promotion ComputerRole will be set to DsRole_RoleBackupDomainController.

The upgrade event is complete when the triggered promotion event is complete. When the upgrade event is complete:

- ComputerUpgrade MUST be set to FALSE.
- PreviousServerState MUST be set to DsRoleServerUnknown.

The operation or set of operations that constitute upgrade are server-to-server operations and are not included in this document; they are not required for interoperation with clients.

### 3.2.5 Message Processing Events and Sequencing Rules

For authenticated **RPC** over **SMB**, the details of method authentication are specific to the underlying RPC implementation, as specified in [C706] section 13, [MS-RPCE] section 5, and [MS-SMB] section 5.

**Opnums** 1 through 11 are not used across the network. These opnums are reserved and MUST NOT be reused by non-Microsoft implementations.<12>

Methods in RPC Opnum Order

| Method | Description |
|---|---|
| DsRolerGetPrimaryDomainInformation | The DsRolerGetPrimaryDomainInformation method returns the requested information about the current configuration or state of the computer on which the server is running.<br>Opnum: 0 |
| Opnum1NotUsedOnWire | Opnum: 1 |
| Opnum2NotUsedOnWire | Opnum: 2 |
| Opnum3NotUsedOnWire | Opnum: 3 |
| Opnum4NotUsedOnWire | Opnum: 4 |
| Opnum5NotUsedOnWire | Opnum: 5 |
| Opnum6NotUsedOnWire | Opnum: 6 |
| Opnum7NotUsedOnWire | Opnum: 7 |
| Opnum8NotUsedOnWire | Opnum: 8 |
| Opnum9NotUsedOnWire | Opnum: 9 |
| Opnum10NotUsedOnWire | Opnum: 10 |
| Opnum11NotUsedOnWire | Opnum: 11 |

All methods MUST NOT throw exceptions.

#### 3.2.5.1 DsRolerGetPrimaryDomainInformation (Opnum 0)

The DsRolerGetPrimaryDomainInformation (Opnum 0) method returns the requested information about the current configuration or state of the computer on which the server is running.

```
DWORD DsRolerGetPrimaryDomainInformation(
  [in] handle t hBinding,
  [in] DSROLE_PRIMARY_DOMAIN_INFO_LEVEL InfoLevel,
  [out, switch is(InfoLevel)] PDSROLER PRIMARY DOMAIN INFORMATION* DomainInfo
);
```

**hBinding:** An RPC binding handle, as specified in [C706] section 2.3.1.

**InfoLevel:** The type of data requested by the client. For possible values in this enumeration, see section 2.2.7.

**DomainInfo:** The requested information that the server provides to the client. The value of the *InfoLevel* parameter indicates the type of information that is requested; information is returned in the corresponding member of the DSROLER_PRIMARY_DOMAIN_INFORMATION union.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code as specified in [MS-ERREF]. Specifically, in addition to any other error codes, the server MUST return the following error codes for the following error conditions. Any other values transmitted in this field are implementation-specific. All nonzero values MUST be treated the same for protocol purposes.

| Return value/code | Description |
|---|---|
| 0x00000057<br>ERROR_INVALID_PARAMETER | One or more parameters are invalid. |
| 0x00000008<br>ERROR_NOT_ENOUGH_MEMORY | A memory allocation failure occurred. |

This method obtains the identity and authorization information about the client from the underlying **RPC** runtime. Servers that implement this method SHOULD impose an authorization policy decision before performing the function.<13>

The server determines the appropriate response to the request by examining the *InfoLevel* parameter, setting the appropriate fields in the *DomainInfo* parameter and sending the response to the caller.

The following describes which fields are used and what the fields contain for each *InfoLevel* value.

**DsRolePrimaryDomainInfoBasic**

When the *InfoLevel* is **DsRolePrimaryDomainInfoBasic**, the server MUST use the **DomainInfoBasic** field of the *DomainInfo* parameter, whose type is DSROLER_PRIMARY_DOMAIN_INFO_BASIC. The result MUST be constructed in the following manner:

1. Determine the role of the server and set the **MachineRole** field of **DomainInfoBasic** according to the ComputerRole state element. If the server ComputerRole state element indicates that it is not a stand-alone computer, set the **DomainNameFlat**, **DomainNameDns**, **DomainForestName**, and **DomainGuid** fields of the DomainInfoBasic structure according to the NetBIOSDomainName, DNSDomainName, ForestName, and DomainGUID state information. If the DomainGUID state element is non-empty, the DSROLE_PRIMARY_DOMAIN_GUID_PRESENT bit MUST be set in the **Flags** member of DomainInfoBasic.

2. If the server is a stand-alone computer, set the **DomainNameFlat** field of DomainInfoBasic according to NetBIOSDomainName state information; and then set the other fields to NULL.

3. If the server is a domain controller and the **directory service** is enabled, set the **Flags** member of the DomainInfoBasic structure as follows:

    1. Set the DSROLE_PRIMARY_DS_RUNNING bit.

    2. If the domain is in mixed mode, set the DSROLE_PRIMARY_DS_MIXED_MODE bit.

    3. If the server is a read-only domain controller, set the DSROLE_PRIMARY_DS_READONLY bit. The domain hosted by a **read-only domain controller** SHOULD<14> be in **native mode**.

**DsRoleUpgradeStatus**

When InfoLevel is DsRoleUpgradeStatus, the server sets the requested information into the **UpgradStatusInfo** field of the *DomainInfo* parameter, whose type is DSROLE_UPGRADE_STATUS_INFO. The result MUST be constructed in the following manner:

1. Set the OperationState field to DSROLE_UPGRADE_IN_PROGRESS if the ComputerUpgrade state element is TRUE.

2. Set the PreviousServerState field to the PreviousServerState state element.

**DsRoleOperationState**

When InfoLevel is DsRoleOperationState, the server MUST return the result in the **OperationStateInfo** field of the *DomainInfo* parameter, whose type is DSROLE_OPERATION_STATE_INFO. The result MUST be constructed by setting the **OperationState** member of the OperationStateInfo structure according to the value of the ComputerOperationState state element.

### 3.2.6 Timer Events

No timer events are required on the server other than the events maintained in the underlying **RPC transport**.

### 3.2.7 Other Local Events

No additional local events are used on the server other than the events maintained in the underlying **RPC transport**.

# 4 Protocol Examples

The following is an example of a DsRolerGetPrimaryDomainInformation **RPC** method.

Assume the server is a workstation computer joined to a domain called MyDomainName.com.

The client calls the DsRolerGetPrimaryDomainInformation RPC method on the server with InfoLevel equal to 1.

The server returns with code 0x00000000; and with the **DomainInfoBasic** field of DomainInfo structure, the following values are in fields of **DomainInfoBasic**.

```
MachineRole = 1
Flags = 0x01000000
DomainNameFlat = "MyDomainName"
DomainNameDns = "MyDomainName.com"
DomainForestName = "MyDomainName.com"
DomainGuid = { 0x5585777b, 0xe549, 0x43b6,
{ 0xa8, 0x42, 0x2, 0xbe, 0xd, 0xd6, 0xab, 0x14 } };
```

# 5 Security

## 5.1 Security Considerations for Implementers

Information returned by this protocol can reveal more than is appropriate for anonymous users, thus resulting in an information leak. An anonymous user can access DsRolerGetPrimaryDomainInformation on a **domain controller** but not on a computer that is not running a domain controller. Implementers therefore need to determine whether to allow access to anonymous users.

## 5.2 Index of Security Parameters

| Security parameter | Section |
|---|---|
| **Remote procedure call (RPC)** authentication. | Section 3.2.5 |
| Allow anonymous users and non-administrative users to retrieve information using the DsRolerGetPrimaryDomainInformation RPC method. | Section 3.2.5.1 |

# 6 Appendix A: Full IDL

```
import "ms-dtyp.idl";
 [
    uuid(3919286a-b10c-11d0-9ba8-00c04fd92ef5),
    version(0.0),
    pointer_default(unique)
]
interface dssetup
{
typedef enum _DSROLE_MACHINE_ROLE {
    DsRole RoleStandaloneWorkstation,
    DsRole_RoleMemberWorkstation,
    DsRole_RoleStandaloneServer,
    DsRole_RoleMemberServer,
    DsRole RoleBackupDomainController,
    DsRole RolePrimaryDomainController
} DSROLE_MACHINE_ROLE;
typedef enum _DSROLE_SERVER_STATE {

    DsRoleServerUnknown = 0,
    DsRoleServerPrimary,
    DsRoleServerBackup
} DSROLE_SERVER_STATE, *PDSROLE_SERVER_STATE;
typedef enum _DSROLE_PRIMARY_DOMAIN_INFO_LEVEL {
    DsRolePrimaryDomainInfoBasic = 1,
    DsRoleUpgradeStatus,
    DsRoleOperationState
} DSROLE PRIMARY DOMAIN INFO LEVEL;
typedef struct _DSROLE_UPGRADE_STATUS_INFO {
    unsigned __int32 OperationState;
    DSROLE_SERVER_STATE PreviousServerState;
} DSROLE UPGRADE STATUS INFO, *PDSROLE UPGRADE STATUS INFO;
typedef enum _DSROLE_OPERATION_STATE {
    DsRoleOperationIdle = 0,
    DsRoleOperationActive,
    DsRoleOperationNeedReboot
} DSROLE_OPERATION_STATE;
typedef struct  DSROLE OPERATION STATE INFO {
    DSROLE OPERATION STATE OperationState;
} DSROLE_OPERATION_STATE_INFO, *PDSROLE_OPERATION_STATE_INFO;

typedef struct _DSROLER_PRIMARY_DOMAIN_INFO_BASIC {
    DSROLE MACHINE ROLE MachineRole;
    unsigned __int32 Flags;
    [ unique, string ] wchar t *DomainNameFlat;
    [ unique, string ] wchar_t *DomainNameDns;
    [ unique, string ] wchar_t *DomainForestName;
    GUID DomainGuid;
} DSROLER PRIMARY DOMAIN INFO BASIC,
 *PDSROLER_PRIMARY_DOMAIN_INFO_BASIC;
typedef [switch type(DSROLE PRIMARY DOMAIN INFO LEVEL)] union
_DSROLER_PRIMARY_DOMAIN_INFORMATION {
    [case(DsRolePrimaryDomainInfoBasic)]
DSROLER_PRIMARY_DOMAIN_INFO_BASIC   DomainInfoBasic;
    [case(DsRoleUpgradeStatus)]
DSROLE_UPGRADE_STATUS_INFO UpgradStatusInfo;
    [case(DsRoleOperationState)]
DSROLE_OPERATION_STATE_INFO OperationStateInfo;
} DSROLER_PRIMARY_DOMAIN_INFORMATION,
*PDSROLER_PRIMARY_DOMAIN_INFORMATION;

DWORD
DsRolerGetPrimaryDomainInformation(
    [in] handle_t hBinding,
    [in] DSROLE_PRIMARY_DOMAIN_INFO_LEVEL InfoLevel,
    [out, switch_is( InfoLevel )]
```

```
PDSROLER_PRIMARY_DOMAIN_INFORMATION *DomainInfo );

/*The following methods are part of the dssetup
interface in Windows 2000, Windows XP RTM,
and Windows XP SP1. They are not part of
this interface in Windows XP SP2 or later
service packs, Windows Server 2003 and later, and Windows Vista and later.
These methods do not expose client server protocol.*/

void Opnum1NotUsedOnWire(void);
void Opnum2NotUsedOnWire(void);
void Opnum3NotUsedOnWire(void);
void Opnum4NotUsedOnWire(void);
void Opnum5NotUsedOnWire(void);
void Opnum6NotUsedOnWire(void);
void Opnum7NotUsedOnWire(void);
void Opnum8NotUsedOnWire(void);
void Opnum9NotUsedOnWire(void);
void Opnum10NotUsedOnWire(void);
void Opnum11NotUsedOnWire(void);
}
```

# 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 updates to those products.

The terms "earlier" and "later", when used with a product version, refer to either all preceding versions or all subsequent versions, respectively. The term "through" refers to the inclusive range of versions. Applicable Microsoft products are listed chronologically in this section.

**Windows Client releases**

- Windows 2000 Professional operating system
- Windows XP operating system
- Windows Vista operating system
- Windows 7 operating system
- Windows 8 operating system
- Windows 8.1 operating system
- Windows 10 operating system

**Windows Server releases**

- Windows 2000 Server operating system
- Windows Server 2003 operating system
- Windows Server 2008 operating system
- Windows Server 2008 R2 operating system
- Windows Server 2012 operating system
- Windows Server 2012 R2 operating system
- Windows Server 2016 operating system
- Windows Server operating system
- Windows Server 2019 operating system

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates 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 1.7: Windows **RPC** protocol returns RPC_S_PROCNUM_OUT_OF_RANGE to notify the client that an RPC method is out of range, as specified in [MS-RPCE].

<2> Section 2.1: Applicable Windows Server releases listen on all protocols bound to RPC. Windows clients attempt only to connect via RPC over the above-named pipe.

<3> Section 2.2.2: **Read-only domain controllers** are not supported in Windows 2000 Server and Windows Server 2003.

<4> Section 2.2.4: In the Windows implementation, after a promotion or demotion operation that requires a reboot, and prior to that reboot, the RPC interface used by this protocol can be unavailable or it can reject connections with authentication errors.

<5> Section 2.2.5: DSROLE_UPGRADE_IN_PROGRESS is only set for an **operating system upgrade** from a Windows NT 4.0 operating system **domain controller**. A Windows computer returns this under the following conditions: (1) it was previously a Windows NT 4.0 domain controller, (2) the operating system upgrade from Windows NT 4.0 has completed, and (3) it has not yet transitioned to being a domain controller.

<6> Section 3.1.3: This protocol configures the RPC runtime to perform a strict **NDR** data consistency check at target level 5.0 for Windows 2000 operating system, Windows XP, and Windows Server 2003, as specified in [MS-RPCE] section 3.

The protocol configures the RPC runtime to perform a strict NDR data consistency check at target level 6.0 for Windows Vista and later and Windows Server 2008 and later.

<7> Section 3.2.1: A Windows **Active Directory domain** has a domain **GUID**, and a Windows NT 4.0 **domain** does not have a domain GUID. Computers running Windows 2000 can be members of a Windows NT 4.0 domain.

<8> Section 3.2.3: This protocol configures the RPC runtime to perform a strict NDR data consistency check at target level 5.0 for Windows 2000, Windows XP, and Windows Server 2003, as specified in [MS-RPCE] section 3.

It configures the RPC runtime to perform a strict NDR data consistency check at target level 6.0 for Windows Vista and later and Windows Server 2008 and later.

In Windows Vista and later and Windows Server 2008 and later, this protocol configures the RPC runtime to reject a NULL unique or full pointer (as specified in [C706] section 14.3.10) with a nonzero conformant value, as specified in [MS-RPCE] section 3.

This protocol configures the RPC runtime via the strict_context_handle attribute to reject the use of context handles that are created by a method of a different RPC interface than this one, as specified in [MS-RPCE] section 3.

<9> Section 3.2.4.1: Windows reinitializes all protocols on the server by rebooting the server.

<10> Section 3.2.4.2: Windows reinitializes all protocols on the server by rebooting the server.

<11> Section 3.2.4.3: Windows only uses a legacy domain as a source for suggested promotion input. Windows allows modification of the suggested input by an administrator before promotion, such as modification of the **NetBIOS name** of the new domain.

<12> Section 3.2.5: Gaps in the **opnum** numbering sequence apply to Windows as follows:

| Opnum | Description |
|---|---|
| 1-11 | Only used locally by Windows, never remotely. |

<13> Section 3.2.5.1: Windows domain controllers allow any authenticated or unauthenticated connection to invoke DsRolerGetPrimaryDomainInformation. Computers running Windows that are not domain controllers require the connection not to be anonymous.

<14> Section 3.2.5.1: Read-only domain controllers are not supported in Windows 2000 Server or Windows Server 2003.

# 8 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

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.
- A document revision that captures changes to protocol functionality.

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 **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

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

| Section | Description | Revision class |
|---|---|---|
| 7 Appendix B: Product Behavior | Added Windows Server 2019 to the list of applicable products. | Major |

# 9 Index

## R



## S



## T



## V