**[MS-DRSR]:**

**Directory Replication Service (DRS) 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](http://go.microsoft.com/fwlink/?LinkId=214445) or the [Microsoft Community Promise](http://go.microsoft.com/fwlink/?LinkId=214448). 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](mailto: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](https://msdn.microsoft.com/en-us/openspecifications/dn750984).
* **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](mailto: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 | Added informative reference; minor updates to content. |
| 8/10/2007 | 1.3.1 | Editorial | Changed language and formatting in the technical content. |
| 9/28/2007 | 1.4 | Minor | Revised references. |
| 10/23/2007 | 2.0 | Major | Updated and revised the technical content. |
| 1/25/2008 | 2.0.1 | Editorial | Changed language and formatting in the technical content. |
| 3/14/2008 | 2.1 | Minor | Clarified the meaning of the technical content. |
| 6/20/2008 | 2.2 | Minor | Clarified the meaning of the technical content. |
| 7/25/2008 | 3.0 | Major | Updated and revised the technical content. |
| 8/29/2008 | 4.0 | Major | Updated and revised the technical content. |
| 10/24/2008 | 5.0 | Major | Updated and revised the technical content. |
| 12/5/2008 | 6.0 | Major | Updated and revised the technical content. |
| 1/16/2009 | 7.0 | Major | Updated and revised the technical content. |
| 2/27/2009 | 8.0 | Major | Updated and revised the technical content. |
| 4/10/2009 | 9.0 | Major | Updated and revised the technical content. |
| 5/22/2009 | 10.0 | Major | Updated and revised the technical content. |
| 7/2/2009 | 11.0 | Major | Updated and revised the technical content. |
| 8/14/2009 | 12.0 | Major | Updated and revised the technical content. |
| 9/25/2009 | 13.0 | Major | Updated and revised the technical content. |
| 11/6/2009 | 14.0 | Major | Updated and revised the technical content. |
| 12/18/2009 | 15.0 | Major | Updated and revised the technical content. |
| 1/29/2010 | 16.0 | Major | Updated and revised the technical content. |
| 3/12/2010 | 17.0 | Major | Updated and revised the technical content. |
| 4/23/2010 | 18.0 | Major | Updated and revised the technical content. |
| 6/4/2010 | 19.0 | Major | Updated and revised the technical content. |
| 7/16/2010 | 20.0 | Major | Updated and revised the technical content. |
| 8/27/2010 | 21.0 | Major | Updated and revised the technical content. |
| 10/8/2010 | 22.0 | Major | Updated and revised the technical content. |
| 11/19/2010 | 23.0 | Major | Updated and revised the technical content. |
| 1/7/2011 | 24.0 | Major | Updated and revised the technical content. |
| 2/11/2011 | 24.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 3/25/2011 | 25.0 | Major | Updated and revised the technical content. |
| 5/6/2011 | 25.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/17/2011 | 25.1 | Minor | Clarified the meaning of the technical content. |
| 9/23/2011 | 26.0 | Major | Updated and revised the technical content. |
| 12/16/2011 | 27.0 | Major | Updated and revised the technical content. |
| 3/30/2012 | 28.0 | Major | Updated and revised the technical content. |
| 7/12/2012 | 29.0 | Major | Updated and revised the technical content. |
| 10/25/2012 | 30.0 | Major | Updated and revised the technical content. |
| 1/31/2013 | 31.0 | Major | Updated and revised the technical content. |
| 8/8/2013 | 32.0 | Major | Updated and revised the technical content. |
| 11/14/2013 | 33.0 | Major | Updated and revised the technical content. |
| 2/13/2014 | 33.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 5/15/2014 | 33.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/30/2015 | 34.0 | Major | Significantly changed the technical content. |
| 10/16/2015 | 34.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/14/2016 | 35.0 | Major | Significantly changed the technical content. |
| 6/1/2017 | 36.0 | Major | Significantly changed the technical content. |
| 9/15/2017 | 37.0 | Major | Significantly changed the technical content. |
| 12/1/2017 | 38.0 | Major | Significantly changed the technical content. |
| 3/16/2018 | 39.0 | Major | Significantly changed the technical content. |

Table of Contents

[1 Introduction 22](#_Toc508101217)

[1.1 Glossary 23](#_Toc508101218)

[1.2 References 35](#_Toc508101219)

[1.2.1 Normative References 35](#_Toc508101220)

[1.2.2 Informative References 37](#_Toc508101221)

[1.3 Overview 37](#_Toc508101222)

[1.3.1 Methods Categorized by Function 37](#_Toc508101223)

[1.3.2 Sequencing Issues 38](#_Toc508101224)

[1.3.3 Most Frequently Used Types 40](#_Toc508101225)

[1.4 Relationship to Other Protocols 41](#_Toc508101226)

[1.5 Prerequisites/Preconditions 41](#_Toc508101227)

[1.6 Applicability Statement 41](#_Toc508101228)

[1.7 Versioning and Capability Negotiation 42](#_Toc508101229)

[1.8 Vendor-Extensible Fields 42](#_Toc508101230)

[1.9 Standards Assignments 42](#_Toc508101231)

[2 Message Transport 43](#_Toc508101232)

[2.1 RPC Transport 43](#_Toc508101233)

[2.2 Protocol Security 43](#_Toc508101234)

[2.2.1 General Background 43](#_Toc508101235)

[2.2.2 Service Principal Names for Domain Controllers 43](#_Toc508101236)

[2.2.3 DC-to-DC Operations 44](#_Toc508101237)

[2.2.3.1 Security Provider 44](#_Toc508101238)

[2.2.3.2 SPN for a Target DC in AD DS 44](#_Toc508101239)

[2.2.3.3 SPN for a Target DC in AD LDS 45](#_Toc508101240)

[2.2.4 Client-to-DC Operations 45](#_Toc508101241)

[2.2.4.1 Security Provider 46](#_Toc508101242)

[2.2.4.2 SPN for a Target DC in AD DS 46](#_Toc508101243)

[2.2.4.3 SPN for a Target DC in AD LDS 47](#_Toc508101244)

[2.3 Directory Service Schema Elements 48](#_Toc508101245)

[3 Background to Behavior Specifications 49](#_Toc508101246)

[3.1 Document Organization 49](#_Toc508101247)

[3.2 Typographical Conventions 49](#_Toc508101248)

[3.3 State Model 49](#_Toc508101249)

[3.3.1 Preliminaries 49](#_Toc508101250)

[3.3.2 Transactions 50](#_Toc508101251)

[3.3.3 Concrete and Abstract Types 50](#_Toc508101252)

[3.4 Pseudocode Language 51](#_Toc508101253)

[3.4.1 Naming Conventions 51](#_Toc508101254)

[3.4.2 Language Constructs for Concrete Types 51](#_Toc508101255)

[3.4.3 Language Constructs for Abstract Types 52](#_Toc508101256)

[3.4.4 Common Language Constructs 54](#_Toc508101257)

[3.4.5 Access to Objects and Their Attributes 55](#_Toc508101258)

[3.4.6 Asynchronous Processing 58](#_Toc508101259)

[3.5 Conventions for Protocol Examples 58](#_Toc508101260)

[3.5.1 Common Configuration 58](#_Toc508101261)

[3.5.2 Data Display Conventions 59](#_Toc508101262)

[3.6 Server and Client Initialization 60](#_Toc508101263)

[3.6.1 AD LDS Specifics 60](#_Toc508101264)

[4 RPC Methods and Their Behavior 61](#_Toc508101265)

[4.1 drsuapi RPC Interface 61](#_Toc508101266)

[4.1.1 IDL\_DRSAddEntry (Opnum 17) 63](#_Toc508101267)

[4.1.1.1 Method-Specific Concrete Types 63](#_Toc508101268)

[4.1.1.1.1 DRS\_MSG\_ADDENTRYREQ 63](#_Toc508101269)

[4.1.1.1.2 DRS\_MSG\_ADDENTRYREQ\_V1 64](#_Toc508101270)

[4.1.1.1.3 DRS\_MSG\_ADDENTRYREQ\_V2 64](#_Toc508101271)

[4.1.1.1.4 DRS\_MSG\_ADDENTRYREQ\_V3 64](#_Toc508101272)

[4.1.1.1.5 DRS\_MSG\_ADDENTRYREPLY 64](#_Toc508101273)

[4.1.1.1.6 DRS\_MSG\_ADDENTRYREPLY\_V1 65](#_Toc508101274)

[4.1.1.1.7 DRS\_MSG\_ADDENTRYREPLY\_V2 65](#_Toc508101275)

[4.1.1.1.8 DRS\_MSG\_ADDENTRYREPLY\_V3 66](#_Toc508101276)

[4.1.1.1.9 ADDENTRY\_REPLY\_INFO 66](#_Toc508101277)

[4.1.1.1.10 DIRERR\_DRS\_WIRE\_V1 67](#_Toc508101278)

[4.1.1.1.11 ATRERR\_DRS\_WIRE\_V1 67](#_Toc508101279)

[4.1.1.1.12 PROBLEMLIST\_DRS\_WIRE\_V1 67](#_Toc508101280)

[4.1.1.1.13 INTFORMPROB\_DRS\_WIRE\_V1 68](#_Toc508101281)

[4.1.1.1.14 NAMERR\_DRS\_WIRE\_V1 68](#_Toc508101282)

[4.1.1.1.15 REFERR\_DRS\_WIRE\_V1 68](#_Toc508101283)

[4.1.1.1.16 NAMERESOP\_DRS\_WIRE\_V1 69](#_Toc508101284)

[4.1.1.1.17 DSA\_ADDRESS\_LIST\_DRS\_WIRE\_V1 69](#_Toc508101285)

[4.1.1.1.18 CONTREF\_DRS\_WIRE\_V1 69](#_Toc508101286)

[4.1.1.1.19 SECERR\_DRS\_WIRE\_V1 70](#_Toc508101287)

[4.1.1.1.20 SVCERR\_DRS\_WIRE\_V1 71](#_Toc508101288)

[4.1.1.1.21 UPDERR\_DRS\_WIRE\_V1 71](#_Toc508101289)

[4.1.1.1.22 SYSERR\_DRS\_WIRE\_V1 71](#_Toc508101290)

[4.1.1.1.23 DRS\_ERROR\_DATA 72](#_Toc508101291)

[4.1.1.1.24 DRS\_ERROR\_DATA\_V1 72](#_Toc508101292)

[4.1.1.1.25 DIRERR Codes 72](#_Toc508101293)

[4.1.1.1.26 PROBLEM Error Codes 73](#_Toc508101294)

[4.1.1.2 Method-Specific Abstract Types and Procedures 75](#_Toc508101295)

[4.1.1.2.1 ConstructReplSpn 75](#_Toc508101296)

[4.1.1.2.2 CreateCrossRef 75](#_Toc508101297)

[4.1.1.2.3 CreateNtdsDsa 77](#_Toc508101298)

[4.1.1.2.4 UseCredsForAccessCheck 78](#_Toc508101299)

[4.1.1.2.5 IsDomainToBeCreated 79](#_Toc508101300)

[4.1.1.2.6 GetDomainNameFromEntinf 79](#_Toc508101301)

[4.1.1.2.7 ENTINF\_GetAttribute 79](#_Toc508101302)

[4.1.1.2.8 SetErrorData 80](#_Toc508101303)

[4.1.1.2.9 ClientIpMatch 80](#_Toc508101304)

[4.1.1.2.10 PerformModifyEntInf 80](#_Toc508101305)

[4.1.1.3 Server Behavior of the IDL\_DRSAddEntry Method 80](#_Toc508101306)

[4.1.2 IDL\_DRSAddSidHistory (Opnum 20) 83](#_Toc508101307)

[4.1.2.1 Method-Specific Concrete Types 84](#_Toc508101308)

[4.1.2.1.1 DRS\_MSG\_ADDSIDREQ 84](#_Toc508101309)

[4.1.2.1.2 DRS\_MSG\_ADDSIDREQ\_V1 84](#_Toc508101310)

[4.1.2.1.3 DRS\_MSG\_ADDSIDREPLY 85](#_Toc508101311)

[4.1.2.1.4 DRS\_MSG\_ADDSIDREPLY\_V1 86](#_Toc508101312)

[4.1.2.1.5 DRS\_ADDSID\_FLAGS 86](#_Toc508101313)

[4.1.2.2 Method-Specific Abstract Types and Procedures 86](#_Toc508101314)

[4.1.2.2.1 ConnectionCtx 86](#_Toc508101315)

[4.1.2.2.2 ConnectToDC 87](#_Toc508101316)

[4.1.2.2.3 ConnectToDCWithCreds 87](#_Toc508101317)

[4.1.2.2.4 GenerateFailureAudit 87](#_Toc508101318)

[4.1.2.2.5 GenerateSuccessAudit 87](#_Toc508101319)

[4.1.2.2.6 GenerateSuccessAuditRemotely 87](#_Toc508101320)

[4.1.2.2.7 GetKeyLength 87](#_Toc508101321)

[4.1.2.2.8 FindGC 87](#_Toc508101322)

[4.1.2.2.9 GetPDC 88](#_Toc508101323)

[4.1.2.2.10 HasAdminRights 88](#_Toc508101324)

[4.1.2.2.11 IsAuditingEnabled 88](#_Toc508101325)

[4.1.2.2.12 IsLocalRpcCall 88](#_Toc508101326)

[4.1.2.2.13 IsNT4SP4OrBetter 88](#_Toc508101327)

[4.1.2.2.14 IsAuditingGroupPresent 88](#_Toc508101328)

[4.1.2.2.15 IsWellKnownDomainRelativeSid 89](#_Toc508101329)

[4.1.2.2.16 LastRID 89](#_Toc508101330)

[4.1.2.2.17 RemoteQuery 89](#_Toc508101331)

[4.1.2.3 Server Behavior of the IDL\_DRSAddSidHistory Method 89](#_Toc508101332)

[4.1.2.4 Examples of the IDL\_DRSAddSidHistory Method 96](#_Toc508101333)

[4.1.2.4.1 Calling IDL\_DRSAddSidHistory with DS\_ADDSID\_FLAG\_PRIVATE\_CHK\_SECURE Flags 96](#_Toc508101334)

[4.1.2.4.1.1 Client Request 96](#_Toc508101335)

[4.1.2.4.1.2 Server Response 97](#_Toc508101336)

[4.1.2.4.1.3 Final State 97](#_Toc508101337)

[4.1.2.4.2 Calling IDL\_DRSAddSidHistory with DS\_ADDSID\_FLAG\_PRIVATE\_DEL\_SRC\_OBJ Flags 97](#_Toc508101338)

[4.1.2.4.2.1 Initial State 97](#_Toc508101339)

[4.1.2.4.2.2 Client Request 98](#_Toc508101340)

[4.1.2.4.2.3 Server Response 98](#_Toc508101341)

[4.1.2.4.2.4 Final State 98](#_Toc508101342)

[4.1.2.4.3 Calling IDL\_DRSAddSidHistory with 0 in Flags 99](#_Toc508101343)

[4.1.2.4.3.1 Initial State 99](#_Toc508101344)

[4.1.2.4.3.2 Client Request 100](#_Toc508101345)

[4.1.2.4.3.3 Server Response 100](#_Toc508101346)

[4.1.2.4.3.4 Final State 100](#_Toc508101347)

[4.1.3 IDL\_DRSBind (Opnum 0) 101](#_Toc508101348)

[4.1.3.1 Client Behavior When Sending the IDL\_DRSBind Request 102](#_Toc508101349)

[4.1.3.2 Server Behavior of the IDL\_DRSBind Method 107](#_Toc508101350)

[4.1.3.3 Client Behavior When Receiving the IDL\_DRSBind Response 111](#_Toc508101351)

[4.1.3.4 Examples of the IDL\_DRSBind Method 111](#_Toc508101352)

[4.1.3.4.1 Initial State 111](#_Toc508101353)

[4.1.3.4.2 Client Request 112](#_Toc508101354)

[4.1.3.4.3 Server Response 113](#_Toc508101355)

[4.1.3.4.4 Final State 114](#_Toc508101356)

[4.1.4 IDL\_DRSCrackNames (Opnum 12) 114](#_Toc508101357)

[4.1.4.1 Method-Specific Concrete Types 115](#_Toc508101358)

[4.1.4.1.1 DRS\_MSG\_CRACKREQ 115](#_Toc508101359)

[4.1.4.1.2 DRS\_MSG\_CRACKREQ\_V1 115](#_Toc508101360)

[4.1.4.1.3 DS\_NAME\_FORMAT 117](#_Toc508101361)

[4.1.4.1.4 DS\_NAME\_RESULT\_ITEMW 118](#_Toc508101362)

[4.1.4.1.5 DS\_NAME\_RESULTW 118](#_Toc508101363)

[4.1.4.1.6 DRS\_MSG\_CRACKREPLY 119](#_Toc508101364)

[4.1.4.1.7 DRS\_MSG\_CRACKREPLY\_V1 119](#_Toc508101365)

[4.1.4.1.8 DS\_NAME\_ERROR 119](#_Toc508101366)

[4.1.4.2 Method-Specific Abstract Types and Procedures 121](#_Toc508101367)

[4.1.4.2.1 CanonicalNameFromCanonicalNameEx 121](#_Toc508101368)

[4.1.4.2.2 DomainDNSNameFromDomain 121](#_Toc508101369)

[4.1.4.2.3 DomainFromDomainDNSName 121](#_Toc508101370)

[4.1.4.2.4 DomainNameFromCanonicalName 121](#_Toc508101371)

[4.1.4.2.5 DomainNameFromSid 121](#_Toc508101372)

[4.1.4.2.6 DomainNameFromUPN 122](#_Toc508101373)

[4.1.4.2.7 DomainNetBIOSNameFromDomain 122](#_Toc508101374)

[4.1.4.2.8 DomainSidFromSid 122](#_Toc508101375)

[4.1.4.2.9 CrackNames 122](#_Toc508101376)

[4.1.4.2.10 LookupName 125](#_Toc508101377)

[4.1.4.2.11 LookupAttr 128](#_Toc508101378)

[4.1.4.2.12 LookupCanonicalName 129](#_Toc508101379)

[4.1.4.2.13 GetCanonicalName 129](#_Toc508101380)

[4.1.4.2.14 LookupSPN 130](#_Toc508101381)

[4.1.4.2.15 LookupSID 130](#_Toc508101382)

[4.1.4.2.16 LookupUnknownName 131](#_Toc508101383)

[4.1.4.2.17 LookupUPNAndAltSecID 131](#_Toc508101384)

[4.1.4.2.18 LookupFPO 132](#_Toc508101385)

[4.1.4.2.19 MapSPN 133](#_Toc508101386)

[4.1.4.2.20 ParseCanonicalName 133](#_Toc508101387)

[4.1.4.2.21 RetrieveDCSuffixFromDn 133](#_Toc508101388)

[4.1.4.2.22 UserNameFromUPN 133](#_Toc508101389)

[4.1.4.2.23 TranslateFPOToName 134](#_Toc508101390)

[4.1.4.2.24 ConstructOutput 134](#_Toc508101391)

[4.1.4.2.25 IsDomainOnly 135](#_Toc508101392)

[4.1.4.3 Server Behavior of the IDL\_DRSCrackNames Method 135](#_Toc508101393)

[4.1.4.4 Examples of the IDL\_DRSCrackNames Method 136](#_Toc508101394)

[4.1.4.4.1 Initial State 136](#_Toc508101395)

[4.1.4.4.2 Client Request 137](#_Toc508101396)

[4.1.4.4.3 Server Response 137](#_Toc508101397)

[4.1.4.4.4 Final State 138](#_Toc508101398)

[4.1.5 IDL\_DRSDomainControllerInfo (Opnum 16) 138](#_Toc508101399)

[4.1.5.1 Method-Specific Concrete Types 138](#_Toc508101400)

[4.1.5.1.1 DRS\_MSG\_DCINFOREQ 138](#_Toc508101401)

[4.1.5.1.2 DRS\_MSG\_DCINFOREQ\_V1 138](#_Toc508101402)

[4.1.5.1.3 DRS\_MSG\_DCINFOREPLY 139](#_Toc508101403)

[4.1.5.1.4 DRS\_MSG\_DCINFOREPLY\_V1 139](#_Toc508101404)

[4.1.5.1.5 DRS\_MSG\_DCINFOREPLY\_V2 139](#_Toc508101405)

[4.1.5.1.6 DRS\_MSG\_DCINFOREPLY\_V3 140](#_Toc508101406)

[4.1.5.1.7 DRS\_MSG\_DCINFOREPLY\_VFFFFFFFF 140](#_Toc508101407)

[4.1.5.1.8 DS\_DOMAIN\_CONTROLLER\_INFO\_1W 140](#_Toc508101408)

[4.1.5.1.9 DS\_DOMAIN\_CONTROLLER\_INFO\_2W 141](#_Toc508101409)

[4.1.5.1.10 DS\_DOMAIN\_CONTROLLER\_INFO\_3W 142](#_Toc508101410)

[4.1.5.1.11 DS\_DOMAIN\_CONTROLLER\_INFO\_FFFFFFFFW 142](#_Toc508101411)

[4.1.5.2 Server Behavior of the IDL\_DRSDomainControllerInfo Method 143](#_Toc508101412)

[4.1.5.3 Examples of the IDL\_DRSDomainControllerInfo Method 147](#_Toc508101413)

[4.1.5.3.1 Initial State 147](#_Toc508101414)

[4.1.5.3.2 Client Request 151](#_Toc508101415)

[4.1.5.3.3 Server Response 151](#_Toc508101416)

[4.1.5.3.4 Final State 152](#_Toc508101417)

[4.1.6 IDL\_DRSExecuteKCC (Opnum 18) 152](#_Toc508101418)

[4.1.6.1 Method-Specific Concrete Types 152](#_Toc508101419)

[4.1.6.1.1 DRS\_MSG\_KCC\_EXECUTE 152](#_Toc508101420)

[4.1.6.1.2 DRS\_MSG\_KCC\_EXECUTE\_V1 152](#_Toc508101421)

[4.1.6.2 Method-Specific Abstract Types and Procedures 153](#_Toc508101422)

[4.1.6.2.1 ExecuteKCCTasks 153](#_Toc508101423)

[4.1.6.3 Server Behavior of the IDL\_DRSExecuteKCC Method 153](#_Toc508101424)

[4.1.7 IDL\_DRSFinishDemotion (Opnum 27) 154](#_Toc508101425)

[4.1.7.1 Method-Specific Concrete Types 154](#_Toc508101426)

[4.1.7.1.1 DRS\_MSG\_FINISH\_DEMOTIONREQ 154](#_Toc508101427)

[4.1.7.1.2 DRS\_MSG\_FINISH\_DEMOTIONREQ\_V1 155](#_Toc508101428)

[4.1.7.1.3 DRS\_MSG\_FINISH\_DEMOTIONREPLY 156](#_Toc508101429)

[4.1.7.1.4 DRS\_MSG\_FINISH\_DEMOTIONREPLY\_V1 156](#_Toc508101430)

[4.1.7.2 Method-Specific Abstract Types and Procedures 156](#_Toc508101431)

[4.1.7.2.1 RemoveADLDSServer 156](#_Toc508101432)

[4.1.7.2.2 RemoveADLDSSCP 157](#_Toc508101433)

[4.1.7.2.3 RemoveADLDSSPNs 157](#_Toc508101434)

[4.1.7.3 Server Behavior of the IDL\_DRSFinishDemotion Method 157](#_Toc508101435)

[4.1.8 IDL\_DRSGetMemberships (Opnum 9) 159](#_Toc508101436)

[4.1.8.1 Method-Specific Concrete Types 160](#_Toc508101437)

[4.1.8.1.1 DRS\_MSG\_REVMEMB\_REQ 160](#_Toc508101438)

[4.1.8.1.2 DRS\_MSG\_REVMEMB\_REQ\_V1 160](#_Toc508101439)

[4.1.8.1.3 REVERSE\_MEMBERSHIP\_OPERATION\_TYPE 160](#_Toc508101440)

[4.1.8.1.4 DRS\_MSG\_REVMEMB\_REPLY 161](#_Toc508101441)

[4.1.8.1.5 DRS\_MSG\_REVMEMB\_REPLY\_V1 161](#_Toc508101442)

[4.1.8.1.6 SE\_GROUP Values 162](#_Toc508101443)

[4.1.8.2 Method-Specific Abstract Types and Procedures 162](#_Toc508101444)

[4.1.8.2.1 Arc and ArcSet 162](#_Toc508101445)

[4.1.8.2.2 Closure 162](#_Toc508101446)

[4.1.8.2.3 DomainOf 163](#_Toc508101447)

[4.1.8.2.4 GetDSNameOfEnterpriseRODCsGroup 163](#_Toc508101448)

[4.1.8.2.5 GetDSNameFromPrimaryGroupId 163](#_Toc508101449)

[4.1.8.2.6 IsMatchedGroup 163](#_Toc508101450)

[4.1.8.2.7 Neighbors 164](#_Toc508101451)

[4.1.8.3 Server Behavior of the IDL\_DRSGetMemberships Method 164](#_Toc508101452)

[4.1.9 IDL\_DRSGetMemberships2 (Opnum 21) 166](#_Toc508101453)

[4.1.9.1 Method-Specific Concrete Types 167](#_Toc508101454)

[4.1.9.1.1 DRS\_MSG\_GETMEMBERSHIPS2\_REQ 167](#_Toc508101455)

[4.1.9.1.2 DRS\_MSG\_GETMEMBERSHIPS2\_REQ\_V1 167](#_Toc508101456)

[4.1.9.1.3 DRS\_MSG\_GETMEMBERSHIPS2\_REPLY 167](#_Toc508101457)

[4.1.9.1.4 DRS\_MSG\_GETMEMBERSHIPS2\_REPLY\_V1 168](#_Toc508101458)

[4.1.9.2 Server Behavior of the IDL\_DRSGetMemberships2 Method 168](#_Toc508101459)

[4.1.10 IDL\_DRSGetNCChanges (Opnum 3) 168](#_Toc508101460)

[4.1.10.1 Overview 169](#_Toc508101461)

[4.1.10.1.1 Cycle Start and Finish 169](#_Toc508101462)

[4.1.10.1.2 Cycle Goal 170](#_Toc508101463)

[4.1.10.1.3 Extended Operations 170](#_Toc508101464)

[4.1.10.2 Method-Specific Concrete Types 171](#_Toc508101465)

[4.1.10.2.1 DRS\_MSG\_GETCHGREQ 171](#_Toc508101466)

[4.1.10.2.2 DRS\_MSG\_GETCHGREQ\_V3 171](#_Toc508101467)

[4.1.10.2.3 DRS\_MSG\_GETCHGREQ\_V4 172](#_Toc508101468)

[4.1.10.2.4 DRS\_MSG\_GETCHGREQ\_V5 173](#_Toc508101469)

[4.1.10.2.5 DRS\_MSG\_GETCHGREQ\_V7 173](#_Toc508101470)

[4.1.10.2.6 DRS\_MSG\_GETCHGREQ\_V8 174](#_Toc508101471)

[4.1.10.2.7 DRS\_MSG\_GETCHGREQ\_V10 175](#_Toc508101472)

[4.1.10.2.8 DRS\_MSG\_GETCHGREQ\_V11 175](#_Toc508101473)

[4.1.10.2.9 DRS\_MSG\_GETCHGREPLY 176](#_Toc508101474)

[4.1.10.2.10 DRS\_MSG\_GETCHGREPLY\_V1 177](#_Toc508101475)

[4.1.10.2.11 DRS\_MSG\_GETCHGREPLY\_V2 178](#_Toc508101476)

[4.1.10.2.12 DRS\_MSG\_GETCHGREPLY\_V6 178](#_Toc508101477)

[4.1.10.2.13 DRS\_MSG\_GETCHGREPLY\_V7 179](#_Toc508101478)

[4.1.10.2.14 DRS\_MSG\_GETCHGREPLY\_V9 179](#_Toc508101479)

[4.1.10.2.15 DRS\_MSG\_GETCHGREPLY\_NATIVE 180](#_Toc508101480)

[4.1.10.2.16 DRS\_MSG\_GETCHGREPLY\_NATIVE\_VERSION\_NUMBER 180](#_Toc508101481)

[4.1.10.2.17 COMPRESSED\_DATA 180](#_Toc508101482)

[4.1.10.2.18 DRS\_COMP\_ALG\_TYPE 180](#_Toc508101483)

[4.1.10.2.19 DRS\_COMPRESSED\_BLOB 181](#_Toc508101484)

[4.1.10.2.20 ENCRYPTED\_PAYLOAD 181](#_Toc508101485)

[4.1.10.2.21 EXOP\_ERR Codes 182](#_Toc508101486)

[4.1.10.2.22 EXOP\_REQ Codes 182](#_Toc508101487)

[4.1.10.2.23 PROPERTY\_META\_DATA 183](#_Toc508101488)

[4.1.10.3 Method-Specific Abstract Types and Procedures 183](#_Toc508101489)

[4.1.10.3.1 AbstractLinkValStampFromConcreteLinkValStamp 183](#_Toc508101490)

[4.1.10.3.2 AbstractPASFromConcretePAS 183](#_Toc508101491)

[4.1.10.3.3 AbstractUTDFromConcreteUTD 184](#_Toc508101492)

[4.1.10.3.4 AttributeAndStamp 184](#_Toc508101493)

[4.1.10.3.5 AttributeStampCompare 184](#_Toc508101494)

[4.1.10.3.6 ConcretePASFromAbstractPAS 184](#_Toc508101495)

[4.1.10.3.7 ConcreteUTDFromAbstractUTD 185](#_Toc508101496)

[4.1.10.3.8 GetNCChangesNativeReply 185](#_Toc508101497)

[4.1.10.3.9 GetStampsForUpdate 186](#_Toc508101498)

[4.1.10.3.10 GetWellKnownObject 186](#_Toc508101499)

[4.1.10.3.11 IsSecretAttribute 187](#_Toc508101500)

[4.1.10.3.12 IsUserIncluded 187](#_Toc508101501)

[4.1.10.3.13 ObjAtts 187](#_Toc508101502)

[4.1.10.3.14 ObjAttVal 187](#_Toc508101503)

[4.1.10.3.15 PerformModifyDNOperation 187](#_Toc508101504)

[4.1.10.3.16 RemoveAttrVal 187](#_Toc508101505)

[4.1.10.3.17 SetAttrStamp 188](#_Toc508101506)

[4.1.10.3.18 SetAttrVal 188](#_Toc508101507)

[4.1.10.3.19 SetLinkStamp 188](#_Toc508101508)

[4.1.10.4 Client Behavior When Sending the IDL\_DRSGetNCChanges Request 188](#_Toc508101509)

[4.1.10.4.1 ReplicateNCRequestMsg 188](#_Toc508101510)

[4.1.10.4.2 ReplSingleObjRequestMsg 192](#_Toc508101511)

[4.1.10.4.3 PerformExtendedOpRequestMsg 193](#_Toc508101512)

[4.1.10.5 Server Behavior of the IDL\_DRSGetNCChanges Method 195](#_Toc508101513)

[4.1.10.5.1 TransformInput 198](#_Toc508101514)

[4.1.10.5.2 GetReplChanges 200](#_Toc508101515)

[4.1.10.5.3 GetReplScope 202](#_Toc508101516)

[4.1.10.5.4 ObjectMatchesSearchFilter 203](#_Toc508101517)

[4.1.10.5.5 GetChangesInScope 203](#_Toc508101518)

[4.1.10.5.6 FilterAttribute 205](#_Toc508101519)

[4.1.10.5.7 GetResponseSubset 206](#_Toc508101520)

[4.1.10.5.8 AddObjToResponse 207](#_Toc508101521)

[4.1.10.5.9 UpdateRevealedList 208](#_Toc508101522)

[4.1.10.5.10 AddLinkToResponse 209](#_Toc508101523)

[4.1.10.5.11 EncryptValuesIfNecessary 210](#_Toc508101524)

[4.1.10.5.12 ProcessFsmoRoleRequest 211](#_Toc508101525)

[4.1.10.5.13 RevealSecretsPolicy 214](#_Toc508101526)

[4.1.10.5.14 GetRevealSecretsPolicyForUser 215](#_Toc508101527)

[4.1.10.5.15 RevealSecretsForUserAllowed 216](#_Toc508101528)

[4.1.10.5.16 GetRoleScope 216](#_Toc508101529)

[4.1.10.5.17 SortResponseLinks 217](#_Toc508101530)

[4.1.10.5.18 ReplValInfV1ListFromReplValInfNativeList 218](#_Toc508101531)

[4.1.10.5.19 ReplValInfNativeListFromReplValInfV1List 218](#_Toc508101532)

[4.1.10.5.20 TransformOutput 219](#_Toc508101533)

[4.1.10.5.21 CompressOrDecompressWin2k3 220](#_Toc508101534)

[4.1.10.5.21.1 LZ77 Compression Algorithm 221](#_Toc508101535)

[4.1.10.5.21.2 DIRECT2 Encoding Algorithm 223](#_Toc508101536)

[4.1.10.5.22 GetOptionalFeatureBit 226](#_Toc508101537)

[4.1.10.6 Client Behavior When Receiving the IDL\_DRSGetNCChanges Response 226](#_Toc508101538)

[4.1.10.6.1 ProcessGetNCChangesReply 226](#_Toc508101539)

[4.1.10.6.2 EnableRecycleBin 229](#_Toc508101540)

[4.1.10.6.3 EnablePrivilegedAccessManagement 229](#_Toc508101541)

[4.1.10.6.4 PrepareCrossNCMove 230](#_Toc508101542)

[4.1.10.6.5 AdjustInstanceTypeAttrVal 231](#_Toc508101543)

[4.1.10.6.6 SetResetInstanceTypeBits 232](#_Toc508101544)

[4.1.10.6.7 PerformModifyOperation 232](#_Toc508101545)

[4.1.10.6.8 NameObject 233](#_Toc508101546)

[4.1.10.6.9 AddObject 234](#_Toc508101547)

[4.1.10.6.10 UpdateObject 235](#_Toc508101548)

[4.1.10.6.11 FindBestParentObject 238](#_Toc508101549)

[4.1.10.6.12 ResolveNameConflict 238](#_Toc508101550)

[4.1.10.6.13 MakeConflictDN 240](#_Toc508101551)

[4.1.10.6.14 ProcessLinkValue 240](#_Toc508101552)

[4.1.10.6.15 UpdateRepsFrom 242](#_Toc508101553)

[4.1.10.6.16 UpdateUTDandPAS 243](#_Toc508101554)

[4.1.10.6.17 DecryptValuesIfNecessary 244](#_Toc508101555)

[4.1.10.6.18 DecompressReplyMessage 245](#_Toc508101556)

[4.1.10.6.19 DecompressMessage 246](#_Toc508101557)

[4.1.10.7 Examples of the IDL\_DRSGetNCChanges Method - Add User 247](#_Toc508101558)

[4.1.10.7.1 Initial State 247](#_Toc508101559)

[4.1.10.7.2 Client Request 249](#_Toc508101560)

[4.1.10.7.3 Server Response 250](#_Toc508101561)

[4.1.10.7.4 Final State 251](#_Toc508101562)

[4.1.10.8 Examples of the IDL\_DRSGetNCChanges Method - Add User to a Group 253](#_Toc508101563)

[4.1.10.8.1 Initial State 253](#_Toc508101564)

[4.1.10.8.2 Client Request 255](#_Toc508101565)

[4.1.10.8.3 Server Response 256](#_Toc508101566)

[4.1.10.8.4 Final State 256](#_Toc508101567)

[4.1.10.9 Examples of the IDL\_DRSGetNCChanges Method - Change User Password 258](#_Toc508101568)

[4.1.10.9.1 Initial State 258](#_Toc508101569)

[4.1.10.9.2 Client Request 260](#_Toc508101570)

[4.1.10.9.3 Server Response 261](#_Toc508101571)

[4.1.10.9.4 Final State 262](#_Toc508101572)

[4.1.11 IDL\_DRSGetNT4ChangeLog (Opnum 11) 263](#_Toc508101573)

[4.1.11.1 Method-Specific Concrete Types 264](#_Toc508101574)

[4.1.11.1.1 DRS\_MSG\_NT4\_CHGLOG\_REQ 264](#_Toc508101575)

[4.1.11.1.2 DRS\_MSG\_NT4\_CHGLOG\_REQ\_V1 264](#_Toc508101576)

[4.1.11.1.3 DRS\_MSG\_NT4\_CHGLOG\_REPLY 265](#_Toc508101577)

[4.1.11.1.4 DRS\_MSG\_NT4\_CHGLOG\_REPLY\_V1 265](#_Toc508101578)

[4.1.11.1.5 NT4\_REPLICATION\_STATE 266](#_Toc508101579)

[4.1.11.2 Method-Specific Abstract Types and Procedures 266](#_Toc508101580)

[4.1.11.2.1 IsPDC 266](#_Toc508101581)

[4.1.11.2.2 GetWindowsErrorCode 266](#_Toc508101582)

[4.1.11.3 Server Behavior of the IDL\_DRSGetNT4ChangeLog Method 267](#_Toc508101583)

[4.1.11.4 Examples of the IDL\_DRSGetNT4ChangeLog Method 270](#_Toc508101584)

[4.1.11.4.1 Initial State 270](#_Toc508101585)

[4.1.11.4.2 Client Request 270](#_Toc508101586)

[4.1.11.4.3 Server Response 270](#_Toc508101587)

[4.1.11.4.4 Final State 270](#_Toc508101588)

[4.1.12 IDL\_DRSGetObjectExistence (Opnum 23) 270](#_Toc508101589)

[4.1.12.1 Method-Specific Concrete Types 271](#_Toc508101590)

[4.1.12.1.1 DRS\_MSG\_EXISTREQ 271](#_Toc508101591)

[4.1.12.1.2 DRS\_MSG\_EXISTREQ\_V1 271](#_Toc508101592)

[4.1.12.1.3 DRS\_MSG\_EXISTREPLY 272](#_Toc508101593)

[4.1.12.1.4 DRS\_MSG\_EXISTREPLY\_V1 272](#_Toc508101594)

[4.1.12.2 Method-Specific Abstract Types and Procedures 272](#_Toc508101595)

[4.1.12.2.1 GuidSequence 272](#_Toc508101596)

[4.1.12.3 Client Behavior When Sending the IDL\_DRSGetObjectExistence Request 273](#_Toc508101597)

[4.1.12.4 Server Behavior of the IDL\_DRSGetObjectExistence Method 274](#_Toc508101598)

[4.1.12.5 Client Behavior When Receiving the IDL\_DRSGetObjectExistence Response 275](#_Toc508101599)

[4.1.13 IDL\_DRSGetReplInfo (Opnum 19) 275](#_Toc508101600)

[4.1.13.1 Method-Specific Concrete Types 276](#_Toc508101601)

[4.1.13.1.1 DRS\_MSG\_GETREPLINFO\_REQ 276](#_Toc508101602)

[4.1.13.1.2 DRS\_MSG\_GETREPLINFO\_REQ\_V1 276](#_Toc508101603)

[4.1.13.1.3 DRS\_MSG\_GETREPLINFO\_REQ\_V2 276](#_Toc508101604)

[4.1.13.1.4 DS\_REPL\_INFO Codes 277](#_Toc508101605)

[4.1.13.1.5 DRS\_MSG\_GETREPLINFO\_REPLY 278](#_Toc508101606)

[4.1.13.1.6 DS\_REPL\_NEIGHBORSW 279](#_Toc508101607)

[4.1.13.1.7 DS\_REPL\_NEIGHBORW 279](#_Toc508101608)

[4.1.13.1.8 DS\_REPL\_CURSORS 280](#_Toc508101609)

[4.1.13.1.9 DS\_REPL\_CURSOR 281](#_Toc508101610)

[4.1.13.1.10 DS\_REPL\_CURSORS\_2 281](#_Toc508101611)

[4.1.13.1.11 DS\_REPL\_CURSOR\_2 281](#_Toc508101612)

[4.1.13.1.12 DS\_REPL\_CURSORS\_3W 282](#_Toc508101613)

[4.1.13.1.13 DS\_REPL\_CURSOR\_3W 282](#_Toc508101614)

[4.1.13.1.14 DS\_REPL\_OBJ\_META\_DATA 282](#_Toc508101615)

[4.1.13.1.15 DS\_REPL\_ATTR\_META\_DATA 283](#_Toc508101616)

[4.1.13.1.16 DS\_REPL\_OBJ\_META\_DATA\_2 283](#_Toc508101617)

[4.1.13.1.17 DS\_REPL\_ATTR\_META\_DATA\_2 284](#_Toc508101618)

[4.1.13.1.18 DS\_REPL\_KCC\_DSA\_FAILURESW 284](#_Toc508101619)

[4.1.13.1.19 DS\_REPL\_KCC\_DSA\_FAILUREW 284](#_Toc508101620)

[4.1.13.1.20 DS\_REPL\_PENDING\_OPSW 285](#_Toc508101621)

[4.1.13.1.21 DS\_REPL\_OPW 285](#_Toc508101622)

[4.1.13.1.22 DS\_REPL\_ATTR\_VALUE\_META\_DATA 286](#_Toc508101623)

[4.1.13.1.23 DS\_REPL\_VALUE\_META\_DATA 286](#_Toc508101624)

[4.1.13.1.24 DS\_REPL\_ATTR\_VALUE\_META\_DATA\_2 287](#_Toc508101625)

[4.1.13.1.25 DS\_REPL\_VALUE\_META\_DATA\_2 287](#_Toc508101626)

[4.1.13.1.26 DS\_REPL\_CLIENT\_CONTEXTS 288](#_Toc508101627)

[4.1.13.1.27 DS\_REPL\_CLIENT\_CONTEXT 288](#_Toc508101628)

[4.1.13.1.28 DS\_REPL\_SERVER\_OUTGOING\_CALLS 289](#_Toc508101629)

[4.1.13.1.29 DS\_REPL\_SERVER\_OUTGOING\_CALL 289](#_Toc508101630)

[4.1.13.2 Method-Specific Abstract Types and Procedures 290](#_Toc508101631)

[4.1.13.2.1 GetDNFromInvocationID 290](#_Toc508101632)

[4.1.13.2.2 GetDNFromObjectGuid 290](#_Toc508101633)

[4.1.13.2.3 GetNCs 291](#_Toc508101634)

[4.1.13.2.4 GetUpToDatenessVector 291](#_Toc508101635)

[4.1.13.3 Server Behavior of the IDL\_DRSGetReplInfo Method 291](#_Toc508101636)

[4.1.13.4 Examples of the IDL\_DRSGetReplInfo Method 302](#_Toc508101637)

[4.1.13.4.1 Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_NEIGHBORS to find replication neighbors for a specified NC 302](#_Toc508101638)

[4.1.13.4.1.1 Initial State 302](#_Toc508101639)

[4.1.13.4.1.2 Client Request 303](#_Toc508101640)

[4.1.13.4.1.3 Server Response 303](#_Toc508101641)

[4.1.13.4.1.4 Final State 304](#_Toc508101642)

[4.1.13.4.2 Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_NEIGHBORS to find the naming contexts for which a DC receives updates from a replication neighbor 304](#_Toc508101643)

[4.1.13.4.2.1 Initial State 304](#_Toc508101644)

[4.1.13.4.2.2 Client Request 305](#_Toc508101645)

[4.1.13.4.2.3 Server Response 306](#_Toc508101646)

[4.1.13.4.2.4 Final State 308](#_Toc508101647)

[4.1.13.4.3 Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_REPSTO to find replication neighbors for a specified NC 308](#_Toc508101648)

[4.1.13.4.3.1 Initial State 308](#_Toc508101649)

[4.1.13.4.3.2 Client Request 308](#_Toc508101650)

[4.1.13.4.3.3 Server Response 309](#_Toc508101651)

[4.1.13.4.3.4 Final State 310](#_Toc508101652)

[4.1.13.4.4 Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_CURSORS\_3\_FOR\_NC 310](#_Toc508101653)

[4.1.13.4.4.1 Initial State 310](#_Toc508101654)

[4.1.13.4.4.2 Client Request 310](#_Toc508101655)

[4.1.13.4.4.3 Server Response 311](#_Toc508101656)

[4.1.13.4.4.4 Final State 311](#_Toc508101657)

[4.1.13.4.5 Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_METADATA\_2\_FOR\_OBJ 312](#_Toc508101658)

[4.1.13.4.5.1 Initial State 312](#_Toc508101659)

[4.1.13.4.5.2 Client Request 312](#_Toc508101660)

[4.1.13.4.5.3 Server Response 313](#_Toc508101661)

[4.1.13.4.5.4 Final State 315](#_Toc508101662)

[4.1.13.4.6 Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_METADATA\_2\_FOR\_ATTR\_VALUE to view the replication metadata for all values of a link value attribute 315](#_Toc508101663)

[4.1.13.4.6.1 Initial State 315](#_Toc508101664)

[4.1.13.4.6.2 Client Request 315](#_Toc508101665)

[4.1.13.4.6.3 Server Response 316](#_Toc508101666)

[4.1.13.4.6.4 Final State 317](#_Toc508101667)

[4.1.13.4.7 Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_METADATA\_2\_FOR\_ATTR\_VALUE to view the replication metadata for a specific value of a link value attribute 318](#_Toc508101668)

[4.1.13.4.7.1 Initial State 318](#_Toc508101669)

[4.1.13.4.7.2 Client Request 318](#_Toc508101670)

[4.1.13.4.7.3 Server Response 318](#_Toc508101671)

[4.1.13.4.7.4 Final State 319](#_Toc508101672)

[4.1.13.4.8 Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_KCC\_DSA\_CONNECT\_FAILURES 319](#_Toc508101673)

[4.1.13.4.8.1 Initial State 319](#_Toc508101674)

[4.1.13.4.8.2 Client Request 319](#_Toc508101675)

[4.1.13.4.8.3 Server Response 320](#_Toc508101676)

[4.1.13.4.8.4 Final State 320](#_Toc508101677)

[4.1.13.4.9 Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_PENDING\_OPS 320](#_Toc508101678)

[4.1.13.4.9.1 Initial State 320](#_Toc508101679)

[4.1.13.4.9.2 Client Request 320](#_Toc508101680)

[4.1.13.4.9.3 Server Response 321](#_Toc508101681)

[4.1.13.4.9.4 Final State 321](#_Toc508101682)

[4.1.14 IDL\_DRSInitDemotion (Opnum 25) 322](#_Toc508101683)

[4.1.14.1 Method-Specific Concrete Types 322](#_Toc508101684)

[4.1.14.1.1 DRS\_MSG\_INIT\_DEMOTIONREQ 322](#_Toc508101685)

[4.1.14.1.2 DRS\_MSG\_INIT\_DEMOTIONREQ\_V1 322](#_Toc508101686)

[4.1.14.1.3 DRS\_MSG\_INIT\_DEMOTIONREPLY 323](#_Toc508101687)

[4.1.14.1.4 DRS\_MSG\_INIT\_DEMOTIONREPLY\_V1 323](#_Toc508101688)

[4.1.14.2 Server Behavior of the IDL\_DRSInitDemotion Method 323](#_Toc508101689)

[4.1.15 IDL\_DRSInterDomainMove (Opnum 10) 324](#_Toc508101690)

[4.1.15.1 Method-Specific Concrete Types 324](#_Toc508101691)

[4.1.15.1.1 DRS\_MSG\_MOVEREQ 324](#_Toc508101692)

[4.1.15.1.2 DRS\_MSG\_MOVEREQ\_V1 325](#_Toc508101693)

[4.1.15.1.3 DRS\_MSG\_MOVEREQ\_V2 325](#_Toc508101694)

[4.1.15.1.4 DRS\_MSG\_MOVEREPLY 325](#_Toc508101695)

[4.1.15.1.5 DRS\_MSG\_MOVEREPLY\_V1 326](#_Toc508101696)

[4.1.15.1.6 DRS\_MSG\_MOVEREPLY\_V2 326](#_Toc508101697)

[4.1.15.2 Method-Specific Abstract Types and Procedures 326](#_Toc508101698)

[4.1.15.2.1 AttrIsBacklink 326](#_Toc508101699)

[4.1.15.2.2 AttrIsConstructed 327](#_Toc508101700)

[4.1.15.2.3 AttrIsNonReplicated 327](#_Toc508101701)

[4.1.15.2.4 AuthorizationInfoFromClientCredentials 327](#_Toc508101702)

[4.1.15.2.5 ImpersonateAuthorizationInfo 327](#_Toc508101703)

[4.1.15.2.6 IsApplicationNC 327](#_Toc508101704)

[4.1.15.2.7 RevertToSelf 327](#_Toc508101705)

[4.1.15.3 Server Behavior of the IDL\_DRSInterDomainMove Method 328](#_Toc508101706)

[4.1.15.4 Examples of the IDL\_DRSInterDomainMove Method 331](#_Toc508101707)

[4.1.15.4.1 Initial State 331](#_Toc508101708)

[4.1.15.4.2 Client Request 332](#_Toc508101709)

[4.1.15.4.3 Server Response 333](#_Toc508101710)

[4.1.15.4.4 Final State 333](#_Toc508101711)

[4.1.16 IDL\_DRSQuerySitesByCost (Opnum 24) 335](#_Toc508101712)

[4.1.16.1 Method-Specific Concrete Types 335](#_Toc508101713)

[4.1.16.1.1 DRS\_MSG\_QUERYSITESREQ 335](#_Toc508101714)

[4.1.16.1.2 DRS\_MSG\_QUERYSITESREQ\_V1 336](#_Toc508101715)

[4.1.16.1.3 DRS\_MSG\_QUERYSITESREPLY 336](#_Toc508101716)

[4.1.16.1.4 DRS\_MSG\_QUERYSITESREPLY\_V1 336](#_Toc508101717)

[4.1.16.1.5 DRS\_MSG\_QUERYSITESREPLYELEMENT\_V1 337](#_Toc508101718)

[4.1.16.2 Method-Specific Abstract Types and Procedures 337](#_Toc508101719)

[4.1.16.2.1 ValidateSiteRDN 337](#_Toc508101720)

[4.1.16.2.2 WeightedArc and WeightedArcSet 337](#_Toc508101721)

[4.1.16.2.3 MinWeightPath 338](#_Toc508101722)

[4.1.16.3 Server Behavior of the IDL\_DRSQuerySitesByCost Method 338](#_Toc508101723)

[4.1.16.4 Examples of IDL\_DRSQuerySitesByCost Method 340](#_Toc508101724)

[4.1.16.4.1 Nontransitive Communication Using siteLinkBridge 340](#_Toc508101725)

[4.1.16.4.1.1 Initial State 341](#_Toc508101726)

[4.1.16.4.1.2 Client Request 347](#_Toc508101727)

[4.1.16.4.1.3 Server Response 347](#_Toc508101728)

[4.1.16.4.1.4 Final State 348](#_Toc508101729)

[4.1.16.4.2 Transitive Communication 348](#_Toc508101730)

[4.1.16.4.2.1 Initial State 349](#_Toc508101731)

[4.1.16.4.2.2 Client Request 353](#_Toc508101732)

[4.1.16.4.2.3 Server Response 354](#_Toc508101733)

[4.1.16.4.2.4 Final State 354](#_Toc508101734)

[4.1.17 IDL\_DRSRemoveDsDomain (Opnum 15) 354](#_Toc508101735)

[4.1.17.1 Method-Specific Concrete Types 355](#_Toc508101736)

[4.1.17.1.1 DRS\_MSG\_RMDMNREQ 355](#_Toc508101737)

[4.1.17.1.2 DRS\_MSG\_RMDMNREQ\_V1 355](#_Toc508101738)

[4.1.17.1.3 DRS\_MSG\_RMDMNREPLY 355](#_Toc508101739)

[4.1.17.1.4 DRS\_MSG\_RMDMNREPLY\_V1 356](#_Toc508101740)

[4.1.17.2 Method-Specific Abstract Types and Procedures 356](#_Toc508101741)

[4.1.17.2.1 HasNCReplicated 356](#_Toc508101742)

[4.1.17.3 Server Behavior of the IDL\_DRSRemoveDsDomain Method 356](#_Toc508101743)

[4.1.18 IDL\_DRSRemoveDsServer (Opnum 14) 357](#_Toc508101744)

[4.1.18.1 Method-Specific Concrete Types 358](#_Toc508101745)

[4.1.18.1.1 DRS\_MSG\_RMSVRREQ 358](#_Toc508101746)

[4.1.18.1.2 DRS\_MSG\_RMSVRREQ\_V1 358](#_Toc508101747)

[4.1.18.1.3 DRS\_MSG\_RMSVRREPLY 359](#_Toc508101748)

[4.1.18.1.4 DRS\_MSG\_RMSVRREPLY\_V1 359](#_Toc508101749)

[4.1.18.2 Server Behavior of the IDL\_DRSRemoveDsServer Method 359](#_Toc508101750)

[4.1.19 IDL\_DRSReplicaAdd (Opnum 5) 362](#_Toc508101751)

[4.1.19.1 Method-Specific Concrete Types 362](#_Toc508101752)

[4.1.19.1.1 DRS\_MSG\_REPADD 362](#_Toc508101753)

[4.1.19.1.2 DRS\_MSG\_REPADD\_V1 363](#_Toc508101754)

[4.1.19.1.3 DRS\_MSG\_REPADD\_V2 363](#_Toc508101755)

[4.1.19.1.4 DRS\_MSG\_REPADD\_V3 363](#_Toc508101756)

[4.1.19.2 Server Behavior of the IDL\_DRSReplicaAdd Method 364](#_Toc508101757)

[4.1.20 IDL\_DRSReplicaDel (Opnum 6) 367](#_Toc508101758)

[4.1.20.1 Method-Specific Concrete Types 368](#_Toc508101759)

[4.1.20.1.1 DRS\_MSG\_REPDEL 368](#_Toc508101760)

[4.1.20.1.2 DRS\_MSG\_REPDEL\_V1 368](#_Toc508101761)

[4.1.20.2 Server Behavior of the IDL\_DRSReplicaDel Method 368](#_Toc508101762)

[4.1.21 IDL\_DRSReplicaDemotion (Opnum 26) 371](#_Toc508101763)

[4.1.21.1 Method-Specific Concrete Types 372](#_Toc508101764)

[4.1.21.1.1 DRS\_MSG\_REPLICA\_DEMOTIONREQ 372](#_Toc508101765)

[4.1.21.1.2 DRS\_MSG\_REPLICA\_DEMOTIONREQ\_V1 372](#_Toc508101766)

[4.1.21.1.3 DRS\_MSG\_REPLICA\_DEMOTIONREPLY 372](#_Toc508101767)

[4.1.21.1.4 DRS\_MSG\_REPLICA\_DEMOTIONREPLY\_V1 373](#_Toc508101768)

[4.1.21.2 Method-Specific Abstract Types and Procedures 373](#_Toc508101769)

[4.1.21.2.1 ReplicationPartners() 373](#_Toc508101770)

[4.1.21.2.2 AbandonAllFSMORoles() 373](#_Toc508101771)

[4.1.21.2.3 ReplicateOffChanges() 374](#_Toc508101772)

[4.1.21.3 Server Behavior of the IDL\_DRSReplicaDemotion Method 375](#_Toc508101773)

[4.1.22 IDL\_DRSReplicaModify (Opnum 7) 376](#_Toc508101774)

[4.1.22.1 Method-Specific Concrete Types 376](#_Toc508101775)

[4.1.22.1.1 DRS\_MSG\_REPMOD 376](#_Toc508101776)

[4.1.22.1.2 DRS\_MSG\_REPMOD\_V1 376](#_Toc508101777)

[4.1.22.2 Server Behavior of the IDL\_DRSReplicaModify Method 377](#_Toc508101778)

[4.1.23 IDL\_DRSReplicaSync (Opnum 2) 379](#_Toc508101779)

[4.1.23.1 Method-Specific Concrete Types 379](#_Toc508101780)

[4.1.23.1.1 DRS\_MSG\_REPSYNC 379](#_Toc508101781)

[4.1.23.1.2 DRS\_MSG\_REPSYNC\_V1 379](#_Toc508101782)

[4.1.23.1.3 DRS\_MSG\_REPSYNC\_V2 380](#_Toc508101783)

[4.1.23.2 Server Behavior of the IDL\_DRSReplicaSync Method 380](#_Toc508101784)

[4.1.24 IDL\_DRSReplicaVerifyObjects (Opnum 22) 382](#_Toc508101785)

[4.1.24.1 Method-Specific Concrete Types 382](#_Toc508101786)

[4.1.24.1.1 DRS\_MSG\_REPVERIFYOBJ 382](#_Toc508101787)

[4.1.24.1.2 DRS\_MSG\_REPVERIFYOBJ\_V1 383](#_Toc508101788)

[4.1.24.2 Method-Specific Abstract Types and Procedures 383](#_Toc508101789)

[4.1.24.2.1 GetRemoteUTD 383](#_Toc508101790)

[4.1.24.2.2 ObjectExistsAtDC 383](#_Toc508101791)

[4.1.24.3 Server Behavior of the IDL\_DRSReplicaVerifyObjects Method 384](#_Toc508101792)

[4.1.24.4 Examples of the IDL\_DRSReplicaVerifyObjects Method 385](#_Toc508101793)

[4.1.24.4.1 Initial State 385](#_Toc508101794)

[4.1.24.4.2 Client Request 388](#_Toc508101795)

[4.1.24.4.3 Server Response 388](#_Toc508101796)

[4.1.24.4.4 Final State 388](#_Toc508101797)

[4.1.25 IDL\_DRSUnbind (Opnum 1) 389](#_Toc508101798)

[4.1.25.1 Server Behavior of the IDL\_DRSUnbind Method 390](#_Toc508101799)

[4.1.26 IDL\_DRSUpdateRefs (Opnum 4) 390](#_Toc508101800)

[4.1.26.1 Method-Specific Concrete Types 390](#_Toc508101801)

[4.1.26.1.1 DRS\_MSG\_UPDREFS 390](#_Toc508101802)

[4.1.26.1.2 DRS\_MSG\_UPDREFS\_V1 391](#_Toc508101803)

[4.1.26.1.3 DRS\_MSG\_UPDREFS\_V2 391](#_Toc508101804)

[4.1.26.2 Server Behavior of the IDL\_DRSUpdateRefs Method 391](#_Toc508101805)

[4.1.26.3 Examples of the IDL\_DRSUpdateRefs Method 392](#_Toc508101806)

[4.1.26.3.1 Adding a repsTo Entry 392](#_Toc508101807)

[4.1.26.3.1.1 Initial State 393](#_Toc508101808)

[4.1.26.3.1.2 Client Request 393](#_Toc508101809)

[4.1.26.3.1.3 Server Response 393](#_Toc508101810)

[4.1.26.3.1.4 Final State 393](#_Toc508101811)

[4.1.26.3.2 Replacing a repsTo Entry 394](#_Toc508101812)

[4.1.26.3.2.1 Initial State 394](#_Toc508101813)

[4.1.26.3.2.2 Client Request 395](#_Toc508101814)

[4.1.26.3.2.3 Server Response 395](#_Toc508101815)

[4.1.26.3.2.4 Final State 395](#_Toc508101816)

[4.1.27 IDL\_DRSVerifyNames (Opnum 8) 396](#_Toc508101817)

[4.1.27.1 Method-Specific Concrete Types 396](#_Toc508101818)

[4.1.27.1.1 DRS\_MSG\_VERIFYREQ 396](#_Toc508101819)

[4.1.27.1.2 DRS\_MSG\_VERIFYREQ\_V1 396](#_Toc508101820)

[4.1.27.1.3 DRS\_MSG\_VERIFYREPLY 397](#_Toc508101821)

[4.1.27.1.4 DRS\_MSG\_VERIFYREPLY\_V1 397](#_Toc508101822)

[4.1.27.2 Server Behavior of the IDL\_DRSVerifyNames Method 398](#_Toc508101823)

[4.1.27.3 Examples of the IDL\_DRSVerifyNames Method 401](#_Toc508101824)

[4.1.27.3.1 Initial State 401](#_Toc508101825)

[4.1.27.3.2 Client Request 401](#_Toc508101826)

[4.1.27.3.3 Server Response 401](#_Toc508101827)

[4.1.27.3.4 Final State 402](#_Toc508101828)

[4.1.28 IDL\_DRSWriteSPN (Opnum 13) 402](#_Toc508101829)

[4.1.28.1 Method-Specific Concrete Types 402](#_Toc508101830)

[4.1.28.1.1 DRS\_MSG\_SPNREQ 402](#_Toc508101831)

[4.1.28.1.2 DRS\_MSG\_SPNREQ\_V1 403](#_Toc508101832)

[4.1.28.1.3 DRS\_MSG\_SPNREPLY 403](#_Toc508101833)

[4.1.28.1.4 DRS\_MSG\_SPNREPLY\_V1 403](#_Toc508101834)

[4.1.28.1.5 DS\_SPN\_OPERATION 404](#_Toc508101835)

[4.1.28.2 Method-Specific Abstract Types and Procedures 404](#_Toc508101836)

[4.1.28.2.1 ExecuteWriteSPNRemotely 404](#_Toc508101837)

[4.1.28.3 Server Behavior of the IDL\_DRSWriteSPN Method 404](#_Toc508101838)

[4.1.29 IDL\_DRSAddCloneDC (Opnum 28) 406](#_Toc508101839)

[4.1.29.1 Method-Specific Concrete Types 407](#_Toc508101840)

[4.1.29.1.1 DRS\_MSG\_ADDCLONEDCREQ 407](#_Toc508101841)

[4.1.29.1.2 DRS\_MSG\_ADDCLONEDCREQ\_V1 407](#_Toc508101842)

[4.1.29.1.3 DRS\_MSG\_ADDCLONEDCREPLY 407](#_Toc508101843)

[4.1.29.1.4 DRS\_MSG\_ADDCLONEDCREPLY\_V1 408](#_Toc508101844)

[4.1.29.2 Method-Specific Abstract Types and Procedures 408](#_Toc508101845)

[4.1.29.2.1 GetKeyLength 408](#_Toc508101846)

[4.1.29.2.2 DNMap 408](#_Toc508101847)

[4.1.29.2.3 DCInfo 408](#_Toc508101848)

[4.1.29.2.4 TranslationInfo 408](#_Toc508101849)

[4.1.29.2.5 ReplaceName 409](#_Toc508101850)

[4.1.29.2.6 ReplaceSIDInSecurityDescriptor 409](#_Toc508101851)

[4.1.29.2.7 GetPrincipalSid 409](#_Toc508101852)

[4.1.29.2.8 GenerateNewKrbTgtAcct 409](#_Toc508101853)

[4.1.29.2.9 DuplicateObject 410](#_Toc508101854)

[4.1.29.3 Server Behavior of the IDL\_DRSAddCloneDC Method 411](#_Toc508101855)

[4.1.29.4 Examples of the IDL\_DRSAddCloneDC Method 415](#_Toc508101856)

[4.1.29.4.1 Initial State 415](#_Toc508101857)

[4.1.29.4.2 Client Request 419](#_Toc508101858)

[4.1.29.4.3 Server Response 419](#_Toc508101859)

[4.1.29.4.4 Final State 419](#_Toc508101860)

[4.1.30 IDL\_DRSWriteNgcKey (Opnum 29) 422](#_Toc508101861)

[4.1.30.1 Method-Specific Concrete Types 423](#_Toc508101862)

[4.1.30.1.1 DRS\_MSG\_WRITENGCKEYREQ 423](#_Toc508101863)

[4.1.30.1.2 DRS\_MSG\_WRITENGCKEYREQ\_V1 423](#_Toc508101864)

[4.1.30.1.3 DRS\_MSG\_WRITENGCKEYREPLY 423](#_Toc508101865)

[4.1.30.1.4 DRS\_MSG\_WRITENGCKEYREPLY\_V1 423](#_Toc508101866)

[4.1.30.2 Method-Specific Abstract Types and Procedures 424](#_Toc508101867)

[4.1.30.2.1 AccessCheckWriteToKeyCredentialLinkAttribute 424](#_Toc508101868)

[4.1.30.2.2 ComposeKeyCredentialLinkForComputer 424](#_Toc508101869)

[4.1.30.3 Server Behavior of the IDL\_DRSWriteNgcKey Method 426](#_Toc508101870)

[4.1.31 IDL\_DRSReadNgcKey (Opnum 30) 427](#_Toc508101871)

[4.1.31.1 Method-Specific Concrete Types 428](#_Toc508101872)

[4.1.31.1.1 DRS\_MSG\_READNGCKEYREQ 428](#_Toc508101873)

[4.1.31.1.2 DRS\_MSG\_READNGCKEYREQ\_V1 428](#_Toc508101874)

[4.1.31.1.3 DRS\_MSG\_READNGCKEYREPLY 428](#_Toc508101875)

[4.1.31.1.4 DRS\_MSG\_READNGCKEYREPLY\_V1 428](#_Toc508101876)

[4.1.31.2 Server Behavior of the IDL\_DRSReadNgcKey Method 429](#_Toc508101877)

[4.2 dsaop RPC Interface 430](#_Toc508101878)

[4.2.1 IDL\_DSAPrepareScript (Opnum 0) 431](#_Toc508101879)

[4.2.1.1 Method-Specific Concrete Types 431](#_Toc508101880)

[4.2.1.1.1 DSA\_MSG\_PREPARE\_SCRIPT\_REQ 431](#_Toc508101881)

[4.2.1.1.2 DSA\_MSG\_PREPARE\_SCRIPT\_REQ\_V1 431](#_Toc508101882)

[4.2.1.1.3 DSA\_MSG\_PREPARE\_SCRIPT\_REPLY 432](#_Toc508101883)

[4.2.1.1.4 DSA\_MSG\_PREPARE\_SCRIPT\_REPLY\_V1 432](#_Toc508101884)

[4.2.1.2 Method-Specific Abstract Types and Procedures 432](#_Toc508101885)

[4.2.1.2.1 GetKeyLengthHandleT 432](#_Toc508101886)

[4.2.1.2.2 PrepareScriptInProgress 432](#_Toc508101887)

[4.2.1.2.3 PrepareScriptVerifyScript 433](#_Toc508101888)

[4.2.1.2.4 PrepareScriptHashBody 433](#_Toc508101889)

[4.2.1.2.5 PrepareScriptHashSignature 433](#_Toc508101890)

[4.2.1.2.6 PrepareScriptGeneratePassword 433](#_Toc508101891)

[4.2.1.3 Server Behavior of the IDL\_DSAPrepareScript Method 433](#_Toc508101892)

[4.2.2 IDL\_DSAExecuteScript (Opnum 1) 435](#_Toc508101893)

[4.2.2.1 Method-Specific Concrete Types 435](#_Toc508101894)

[4.2.2.1.1 DSA\_MSG\_EXECUTE\_SCRIPT\_REQ 435](#_Toc508101895)

[4.2.2.1.2 DSA\_MSG\_EXECUTE\_SCRIPT\_REQ\_V1 436](#_Toc508101896)

[4.2.2.1.3 DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY 436](#_Toc508101897)

[4.2.2.1.4 DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY\_V1 436](#_Toc508101898)

[4.2.2.2 Method-Specific Abstract Types and Procedures 436](#_Toc508101899)

[4.2.2.2.1 ExecuteScriptInProgress 436](#_Toc508101900)

[4.2.2.2.2 ExecuteScript 437](#_Toc508101901)

[4.2.2.3 Server Behavior of the IDL\_DSAExecuteScript Method 437](#_Toc508101902)

[5 Common Data Types, Variables, and Procedures 439](#_Toc508101903)

[5.1 AbstractPTFromConcretePT 439](#_Toc508101904)

[5.2 AccessCheckAttr 439](#_Toc508101905)

[5.3 AccessCheckCAR 440](#_Toc508101906)

[5.4 AccessCheckObject 440](#_Toc508101907)

[5.5 AccessCheckWriteToSpnAttribute 440](#_Toc508101908)

[5.6 AddSubRef 441](#_Toc508101909)

[5.7 AmIRODC 442](#_Toc508101910)

[5.8 AmILHServer 442](#_Toc508101911)

[5.9 ATTR 442](#_Toc508101912)

[5.10 ATTRBLOCK 443](#_Toc508101913)

[5.11 AttributeStamp 443](#_Toc508101914)

[5.12 AttributeSyntax 444](#_Toc508101915)

[5.13 AttrStamp 444](#_Toc508101916)

[5.14 ATTRTYP 445](#_Toc508101917)

[5.15 AttrtypFromSchemaObj 445](#_Toc508101918)

[5.16 ATTRVAL 445](#_Toc508101919)

[5.16.1 Concrete Value Representations 445](#_Toc508101920)

[5.16.1.1 INT32 446](#_Toc508101921)

[5.16.1.2 INT64 446](#_Toc508101922)

[5.16.1.3 OctetString 446](#_Toc508101923)

[5.16.1.4 String8 446](#_Toc508101924)

[5.16.1.5 String16 446](#_Toc508101925)

[5.16.1.6 SECURITY\_DESCRIPTOR 446](#_Toc508101926)

[5.16.1.7 SID 446](#_Toc508101927)

[5.16.2 Abstract Value Representations 447](#_Toc508101928)

[5.16.2.1 Object(DS-DN) 447](#_Toc508101929)

[5.16.2.2 Object(DN-String) 448](#_Toc508101930)

[5.16.2.3 Object(DN-Binary) 448](#_Toc508101931)

[5.16.2.4 Object(Access-Point) 448](#_Toc508101932)

[5.16.2.5 Object(OR-Name) 449](#_Toc508101933)

[5.16.2.6 String(NT-Sec-Desc) 449](#_Toc508101934)

[5.16.2.7 String(Sid) 449](#_Toc508101935)

[5.16.2.8 String(Teletex) 449](#_Toc508101936)

[5.16.3 Converting Between Abstract and Concrete Value Representations 449](#_Toc508101937)

[5.16.3.1 Boolean 451](#_Toc508101938)

[5.16.3.2 Enumeration and Integer 451](#_Toc508101939)

[5.16.3.3 LargeInteger 452](#_Toc508101940)

[5.16.3.4 Object(Presentation-Address) 452](#_Toc508101941)

[5.16.3.5 Object(Replica-Link) String (Octet) 452](#_Toc508101942)

[5.16.3.6 String(IA5) String(Printable) String(Numeric) String(Teletex) 452](#_Toc508101943)

[5.16.3.7 String(Unicode) 453](#_Toc508101944)

[5.16.3.8 String(Object-Identifier) 453](#_Toc508101945)

[5.16.3.9 String(UTC-Time) and String(Generalized-Time) 453](#_Toc508101946)

[5.16.3.10 Object(DS-DN) 454](#_Toc508101947)

[5.16.3.11 Object(DN-Binary) 455](#_Toc508101948)

[5.16.3.12 Object(DN-String) 456](#_Toc508101949)

[5.16.3.13 Object(OR-Name) 457](#_Toc508101950)

[5.16.3.14 Object(Access-Point) 457](#_Toc508101951)

[5.16.3.15 String(Sid) 457](#_Toc508101952)

[5.16.3.16 String(NT-Sec-Desc) 457](#_Toc508101953)

[5.16.4 ATTRTYP-to-OID Conversion 458](#_Toc508101954)

[5.17 ATTRVALBLOCK 464](#_Toc508101955)

[5.18 ATTRVALFromValue 464](#_Toc508101956)

[5.19 BindToDSA() 464](#_Toc508101957)

[5.20 BOOL 464](#_Toc508101958)

[5.21 BYTE 464](#_Toc508101959)

[5.22 CHANGE\_LOG\_ENTRIES 464](#_Toc508101960)

[5.23 CHANGELOG\_ENTRY 465](#_Toc508101961)

[5.24 CheckGroupMembership 465](#_Toc508101962)

[5.25 ClientAuthorizationInfo 465](#_Toc508101963)

[5.26 ClientExtensions 465](#_Toc508101964)

[5.27 ClientUUID 465](#_Toc508101965)

[5.28 ConcretePTFromAbstractPT 466](#_Toc508101966)

[5.29 ConfigNC 466](#_Toc508101967)

[5.30 dc, DC 466](#_Toc508101968)

[5.31 DefaultNC 467](#_Toc508101969)

[5.32 DelSubRef 467](#_Toc508101970)

[5.33 DescendantObject 468](#_Toc508101971)

[5.34 DomainNameFromDN 468](#_Toc508101972)

[5.35 DN 468](#_Toc508101973)

[5.36 DNBinary 468](#_Toc508101974)

[5.37 DomainNameFromNT4AccountName 468](#_Toc508101975)

[5.38 DRS\_EXTENSIONS 468](#_Toc508101976)

[5.39 DRS\_EXTENSIONS\_INT 469](#_Toc508101977)

[5.40 DRS\_HANDLE 472](#_Toc508101978)

[5.41 DRS\_OPTIONS 473](#_Toc508101979)

[5.42 DRS\_MORE\_GETCHGREQ\_OPTIONS 475](#_Toc508101980)

[5.43 DRS\_SecBuffer 475](#_Toc508101981)

[5.44 DRS\_SecBufferDesc 477](#_Toc508101982)

[5.45 DRS\_SPN\_CLASS 477](#_Toc508101983)

[5.46 DS\_REPL\_OP\_TYPE 477](#_Toc508101984)

[5.47 DSAObj 477](#_Toc508101985)

[5.48 DSA\_RPC\_INST 478](#_Toc508101986)

[5.49 DSName 478](#_Toc508101987)

[5.50 DSNAME 478](#_Toc508101988)

[5.50.1 DSNAME Equality 480](#_Toc508101989)

[5.51 DSTIME 480](#_Toc508101990)

[5.52 DWORD 481](#_Toc508101991)

[5.53 ENTINF 481](#_Toc508101992)

[5.54 ENTINF\_GetValue 481](#_Toc508101993)

[5.55 ENTINF\_SetValue 482](#_Toc508101994)

[5.56 ENTINF\_EnumerateAttributes 482](#_Toc508101995)

[5.57 ENTINFLIST 482](#_Toc508101996)

[5.58 Expunge 483](#_Toc508101997)

[5.59 FILETIME 483](#_Toc508101998)

[5.60 FilteredGCPAS 483](#_Toc508101999)

[5.61 FilteredPAS 483](#_Toc508102000)

[5.62 FindChar 484](#_Toc508102001)

[5.63 FindCharRev 484](#_Toc508102002)

[5.64 FOREST\_TRUST\_INFORMATION 485](#_Toc508102003)

[5.64.1 Record 485](#_Toc508102004)

[5.64.2 Determining If a Name Is in a Trusted Forest 488](#_Toc508102005)

[5.65 FOREST\_TRUST\_RECORD\_TYPE 494](#_Toc508102006)

[5.66 ForestRootDomainNC 495](#_Toc508102007)

[5.67 FullReplicaExists 495](#_Toc508102008)

[5.68 GCPAS 495](#_Toc508102009)

[5.69 GetFilteredAttributeSet 495](#_Toc508102010)

[5.70 GetNCType 496](#_Toc508102011)

[5.71 GetAttrVals 497](#_Toc508102012)

[5.72 GetCallerAuthorizationInfo 497](#_Toc508102013)

[5.73 GetDefaultObjectCategory 497](#_Toc508102014)

[5.74 GetDomainNC 497](#_Toc508102015)

[5.75 GetDSNameFromAttrVal 498](#_Toc508102016)

[5.76 GetDSNameFromDN 498](#_Toc508102017)

[5.77 GetDSNameFromNetworkAddress 498](#_Toc508102018)

[5.78 GetForestFunctionalLevel 498](#_Toc508102019)

[5.79 GetFSMORoleOwner 498](#_Toc508102020)

[5.80 GetInstanceNameFromSPN 499](#_Toc508102021)

[5.81 GetObjectNC 499](#_Toc508102022)

[5.82 GetProxyEpoch 499](#_Toc508102023)

[5.83 GetProxyType 499](#_Toc508102024)

[5.84 GetServiceClassFromSPN 499](#_Toc508102025)

[5.85 GetServiceNameFromSPN 500](#_Toc508102026)

[5.86 groupType Bit Flags 500](#_Toc508102027)

[5.87 GUID 500](#_Toc508102028)

[5.88 GuidFromString 501](#_Toc508102029)

[5.89 GuidToString 501](#_Toc508102030)

[5.90 handle\_t 501](#_Toc508102031)

[5.91 instanceType Bit Flags 501](#_Toc508102032)

[5.92 Is2PartSPN 502](#_Toc508102033)

[5.93 Is3PartSPN 502](#_Toc508102034)

[5.94 IsAdlds 502](#_Toc508102035)

[5.95 IsBuiltinPrincipal 502](#_Toc508102036)

[5.96 IsDomainNameInTrustedForest 502](#_Toc508102037)

[5.97 IsDomainSidInTrustedForest 502](#_Toc508102038)

[5.98 IsDCAccount 503](#_Toc508102039)

[5.99 IsForwardLinkAttribute 503](#_Toc508102040)

[5.100 IsGC 503](#_Toc508102041)

[5.101 IsGetNCChangesPermissionGranted 503](#_Toc508102042)

[5.102 IsGUIDBasedDNSName 504](#_Toc508102043)

[5.103 IsMemberOfBuiltinAdminGroup 504](#_Toc508102044)

[5.104 IsRecycleBinEnabled 504](#_Toc508102045)

[5.105 IsRevealFilteredAttribute 504](#_Toc508102046)

[5.106 IsPrivilegedAccessManagementEnabled 505](#_Toc508102047)

[5.107 IsRevealSecretRequest 505](#_Toc508102048)

[5.108 IsServerExtensionsChanged 506](#_Toc508102049)

[5.109 IsUPNInTrustedForest 506](#_Toc508102050)

[5.110 IsValidServiceName 506](#_Toc508102051)

[5.111 KCCFailedConnections 507](#_Toc508102052)

[5.112 KCCFailedLinks 507](#_Toc508102053)

[5.113 LARGE\_INTEGER 507](#_Toc508102054)

[5.114 LDAP\_CONN\_PROPERTIES 507](#_Toc508102055)

[5.115 LDAP\_SERVER\_DIRSYNC\_OID LDAP Search Control 508](#_Toc508102056)

[5.115.1 Abstract Types 508](#_Toc508102057)

[5.115.1.1 AttributeList 508](#_Toc508102058)

[5.115.1.2 AttributeListElement 508](#_Toc508102059)

[5.115.1.3 AttributeVals 509](#_Toc508102060)

[5.115.1.4 Control 509](#_Toc508102061)

[5.115.1.5 DirSyncControlValue 509](#_Toc508102062)

[5.115.1.6 DirSyncSearchArg 509](#_Toc508102063)

[5.115.1.7 LDAPString 510](#_Toc508102064)

[5.115.1.8 SearchResultEntry 510](#_Toc508102065)

[5.115.1.9 SearchResultEntryList 510](#_Toc508102066)

[5.115.2 Concrete Types 510](#_Toc508102067)

[5.115.2.1 Cookie 510](#_Toc508102068)

[5.115.3 ProcessDirSyncSearchRequest 511](#_Toc508102069)

[5.115.4 DirSyncReqToGetChgReq 512](#_Toc508102070)

[5.115.5 GetChgReplyToSearchResult 513](#_Toc508102071)

[5.115.6 TransformEntinfToSearchEntry 514](#_Toc508102072)

[5.115.7 TransformReplValInfNativeListToSearchEntry 515](#_Toc508102073)

[5.115.8 TransformDSNameToLdapDN 516](#_Toc508102074)

[5.115.9 LDAPDisplayNameFromAttrTyp 516](#_Toc508102075)

[5.115.10 GetResponseDirSyncControlValue 516](#_Toc508102076)

[5.115.11 GetUsnUtdVectorFromCookie 517](#_Toc508102077)

[5.115.12 SecurityCheckForChanges 518](#_Toc508102078)

[5.115.13 IsFilteredAttributePresent 519](#_Toc508102079)

[5.116 LDAPConnections 519](#_Toc508102080)

[5.117 LinkStamp 520](#_Toc508102081)

[5.118 LinkValueStamp 520](#_Toc508102082)

[5.119 LinkValueStampCompare 521](#_Toc508102083)

[5.120 LocalAttidFromRemoteAttid 522](#_Toc508102084)

[5.121 LONG 522](#_Toc508102085)

[5.122 LONGLONG 522](#_Toc508102086)

[5.123 LPWSTR 522](#_Toc508102087)

[5.124 MakeAttid 522](#_Toc508102088)

[5.125 MakeProxyValue 522](#_Toc508102089)

[5.126 MasterReplicaExists 522](#_Toc508102090)

[5.127 MD5\_CTX 523](#_Toc508102091)

[5.128 MD5Final 523](#_Toc508102092)

[5.129 MD5Init 523](#_Toc508102093)

[5.130 MD5Update 523](#_Toc508102094)

[5.131 MergeUTD 523](#_Toc508102095)

[5.132 MTX\_ADDR 523](#_Toc508102096)

[5.133 NCType Bits 524](#_Toc508102097)

[5.134 NetworkAddress 524](#_Toc508102098)

[5.135 NewPrefixTable 525](#_Toc508102099)

[5.136 Nt4ReplicationState 525](#_Toc508102100)

[5.137 NT4SID 525](#_Toc508102101)

[5.138 NTSAPI\_CLIENT\_GUID 526](#_Toc508102102)

[5.139 NTDSTRANSPORT\_OPT Values 526](#_Toc508102103)

[5.140 NULLGUID 526](#_Toc508102104)

[5.141 ObjExists 526](#_Toc508102105)

[5.142 OID 526](#_Toc508102106)

[5.143 OID\_t 526](#_Toc508102107)

[5.144 OidFromAttid 526](#_Toc508102108)

[5.145 parent 527](#_Toc508102109)

[5.146 PARTIAL\_ATTR\_VECTOR\_V1\_EXT 527](#_Toc508102110)

[5.147 partialAttributeSet 527](#_Toc508102111)

[5.148 PartialGCReplicaExists 527](#_Toc508102112)

[5.149 PAS\_DATA 527](#_Toc508102113)

[5.150 PdcChangeLog 528](#_Toc508102114)

[5.151 PerformAddOperation 528](#_Toc508102115)

[5.152 PerformAddOperationAsSystem 529](#_Toc508102116)

[5.153 PrefixTable 529](#_Toc508102117)

[5.154 PrefixTableEntry 529](#_Toc508102118)

[5.155 PROPERTY\_META\_DATA\_EXT 529](#_Toc508102119)

[5.156 PROPERTY\_META\_DATA\_EXT\_VECTOR 530](#_Toc508102120)

[5.157 proxiedObjectName Value Format 530](#_Toc508102121)

[5.158 RDN 530](#_Toc508102122)

[5.159 rdnType 530](#_Toc508102123)

[5.160 RecycleObj 530](#_Toc508102124)

[5.161 RemoveObj 531](#_Toc508102125)

[5.162 REPLENTINFLIST 531](#_Toc508102126)

[5.163 ReplicatedAttributes 531](#_Toc508102127)

[5.164 ReplicationQueue 532](#_Toc508102128)

[5.165 REPLTIMES 532](#_Toc508102129)

[5.166 replUpToDateVector, ReplUpToDateVector 533](#_Toc508102130)

[5.167 REPLVALINF\_V1 533](#_Toc508102131)

[5.168 REPLVALINF\_V3 534](#_Toc508102132)

[5.169 REPLVALINF\_NATIVE 534](#_Toc508102133)

[5.170 REPS\_FROM 534](#_Toc508102134)

[5.171 REPS\_TO 536](#_Toc508102135)

[5.172 repsFrom, RepsFrom 539](#_Toc508102136)

[5.173 repsTo, RepsTo 540](#_Toc508102137)

[5.174 Rid 542](#_Toc508102138)

[5.175 Right 542](#_Toc508102139)

[5.176 RIGHT Values 542](#_Toc508102140)

[5.177 RPCClientContexts 542](#_Toc508102141)

[5.178 RPCOutgoingContexts 543](#_Toc508102142)

[5.179 sAMAccountType Values 543](#_Toc508102143)

[5.180 SCHEMA\_PREFIX\_TABLE 544](#_Toc508102144)

[5.181 SchemaInfo 544](#_Toc508102145)

[5.182 SchemaNC 544](#_Toc508102146)

[5.183 SchemaObj 544](#_Toc508102147)

[5.184 ServerExtensions 544](#_Toc508102148)

[5.185 SID 545](#_Toc508102149)

[5.186 SidFromStringSid 545](#_Toc508102150)

[5.187 StampLessThanOrEqualUTD 545](#_Toc508102151)

[5.188 StartsWith 545](#_Toc508102152)

[5.189 StringSidFromSid 545](#_Toc508102153)

[5.190 SubString 545](#_Toc508102154)

[5.191 Syntax 546](#_Toc508102155)

[5.192 SYNTAX\_ADDRESS 546](#_Toc508102156)

[5.193 SYNTAX\_DISTNAME\_BINARY 546](#_Toc508102157)

[5.194 systemFlags Values 548](#_Toc508102158)

[5.195 UCHAR 548](#_Toc508102159)

[5.196 ULARGE\_INTEGER 548](#_Toc508102160)

[5.197 ULONG 548](#_Toc508102161)

[5.198 ULONGLONG 548](#_Toc508102162)

[5.199 UndeleteObject 548](#_Toc508102163)

[5.200 UnbindFromDSA() 548](#_Toc508102164)

[5.201 UpdateRefs 549](#_Toc508102165)

[5.202 UPTODATE\_CURSOR\_V1 549](#_Toc508102166)

[5.203 UPTODATE\_CURSOR\_V2 550](#_Toc508102167)

[5.204 UPTODATE\_VECTOR\_V1\_EXT 550](#_Toc508102168)

[5.205 UPTODATE\_VECTOR\_V2\_EXT 551](#_Toc508102169)

[5.206 userAccountControl Bits 551](#_Toc508102170)

[5.207 UserNameFromNT4AccountName 552](#_Toc508102171)

[5.208 USHORT 552](#_Toc508102172)

[5.209 USN 552](#_Toc508102173)

[5.210 USN\_VECTOR 552](#_Toc508102174)

[5.211 UUID 552](#_Toc508102175)

[5.212 ValidateDRSDemotionInput 552](#_Toc508102176)

[5.213 ValidateDRSInput 553](#_Toc508102177)

[5.214 Value 554](#_Toc508102178)

[5.215 VALUE\_META\_DATA\_EXT\_V1 554](#_Toc508102179)

[5.216 VALUE\_META\_DATA\_EXT\_V3 554](#_Toc508102180)

[5.217 VALUE\_META\_DATA\_EXT\_NATIVE 554](#_Toc508102181)

[5.218 ValueFromATTRVAL 555](#_Toc508102182)

[5.219 VAR\_SIZE\_BUFFER\_WITH\_VERSION 555](#_Toc508102183)

[5.220 WCHAR 555](#_Toc508102184)

[6 Security 556](#_Toc508102185)

[6.1 Security Considerations for Implementers 556](#_Toc508102186)

[6.2 Index of Security Parameters 556](#_Toc508102187)

[7 Appendix A: Full IDL 557](#_Toc508102188)

[8 Appendix B: Product Behavior 584](#_Toc508102189)

[9 Change Tracking 592](#_Toc508102190)

[10 Index 594](#_Toc508102191)

# Introduction

The Directory Replication Service (DRS) Remote Protocol is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol for [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) and management of data in [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90).

The protocol consists of two RPC interfaces named drsuapi and dsaop. The name of each drsuapi method begins with "IDL\_DRS", while the name of each dsaop method begins with "IDL\_DSA". This protocol was originally implemented in Windows 2000 Server operating system and is available in all subsequent server releases. It is not available in Windows NT 3.1 operating system, Windows NT 3.51 operating system, or Windows NT 4.0 operating system.

Some functionality exposed by these RPC protocols is also available using the [**Lightweight Directory Access Protocol (LDAP)**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) protocol ([[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.3); the overlap is described in section [1.4](#Section_0597aff601774d5299f214a5441bc3c1).

The special typographical conventions used in this document are described in section [3.2](#Section_d05f9217c26b4f64b16e2247389319ff).

State is included in the state model for this specification only as necessitated by the requirement that a licensee implementation of the protocols of applicable Windows Server releases has to be capable of receiving messages and responding in the same manner as applicable Windows Server releases. Behavior is specified in terms of request message received, processing based on current state, resulting state transformation, and response message sent. Unless otherwise specified, all behaviors are required elements of the protocol. Any specified behavior not explicitly qualified with MAY or SHOULD is to be treated as if it were specified as a MUST behavior.

**AD LDS for Windows Client operating systems**

Note that information that is applicable to Active Directory Lightweight Directory Services (AD LDS) on applicable Windows Server releases is also generally applicable to AD LDS on Windows clients. For more information, see [MS-ADTS] section 1.

**Pervasive Concepts**

The following concepts are pervasive throughout this specification.

This specification uses [KNUTH1] section 2.3.4.2 as a reference for the graph-related terms [**oriented tree**](#gt_c62abd7d-b916-4873-9251-7781d497af27), root, vertex, arc, initial vertex, and final vertex.

**replica:** A variable containing a set of objects.

**attribute:** An identifier for a value or set of values. See also attribute in the [Glossary (section 1.1)](#Section_e5c2026bf7324c9d9d60b945c0ab54eb).

**object:** A set of attributes, each with its associated values. Two attributes of an object have special significance:

* Identifying attribute: A designated single-valued attribute appears on every object. The value of this attribute identifies the object. For the set of objects in a replica, the values of the identifying attribute are distinct.
* Parent-identifying attribute: A designated single-valued attribute appears on every object. The value of this attribute identifies the [**object's**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) parent. That is, this attribute contains the value of the parent's identifying attribute or a reserved value identifying no object (for more information, see [MS-ADTS] section 3.1.1.1.3). For the set of objects in a replica, the values of this parent-identifying attribute define an oriented tree with objects as vertices and child-parent references as directed arcs, with the child as an arc's initial vertex and the parent as an arc's final vertex.

Note that an object is a value, not a variable; a [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) is a variable. The process of adding, modifying, or deleting an object in a replica replaces the entire value of the replica with a new value.

As the term "replica" suggests, it is often the case that two replicas contain "the same objects". In this usage, objects in two replicas are considered "the same" if they have the same value of the identifying attribute and if there is a process in place (that is, replication) to converge the values of the remaining attributes. When the members of a set of replicas are considered to be the same, it is common to say "an object" as a shorthand way of referring to the set of corresponding objects in the replicas.

**object class:** A set of restrictions on the construction and [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) of objects. An object class must be specified when an object is created. An object class specifies a set of must-have attributes (every object of the class must have at least one value of each) and may-have attributes (every object of the class may have a value of each). An object class also specifies a set of possible superiors (the parent object of an object of the class must have one of these classes). An object class is defined by a classSchema object.

**parent object:** See "object", above.

**child object, children:** An object that is not the root of its oriented tree. The children of an object *O* is the set of all objects whose parent is *O*.

See [MS-ADTS] section 3.1.1.1.3 for the particular use made of these definitions in this specification.

Sections [1.5](#Section_c3379fc5123d4f28922d97308561d748), [1.8](#Section_9b545b76247c44aaba77a3581fa4205a), [1.9](#Section_063618edb2e24983ab133ed056700641), [2](#Section_d8a5de321b81441cb3bcce90b1ccd178), [3](#Section_c56432ffaf884443b500eecb3047da4d), [4](#Section_9554afa5e7554742a34b899fc4e2fd20), and [5](#Section_c5d9026516534ecca0d7cac691e2d08e) of this specification are normative. All other sections and examples in this specification are informative.

## Glossary

This document uses the following terms:

**abstract type**: A type used in this specification whose representation need not be standardized for interoperability because the type's use is internal to the specification. See also [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d).

**access control list (ACL)**: A list of access control entries (ACEs) that collectively describe the security rules for authorizing access to some resource; for example, an object or set of objects.

**Active Directory**: A general-purpose network directory service. [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) also refers to the Windows implementation of a directory service. [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) stores information about a variety of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the network. User accounts, computer accounts, groups, and all related credential information used by the Windows implementation of Kerberos are stored in [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90). [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) is either deployed as [**Active Directory Domain Services (AD DS)**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024) or [**Active Directory Lightweight Directory Services (AD LDS)**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab). [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) describes both forms. For more information, see [[MS-AUTHSOD]](%5bMS-AUTHSOD%5d.pdf#Section_953d700a57cb4cf7b0c3a64f34581cc9) section 1.1.1.5.2, [**Lightweight Directory Access Protocol (LDAP)**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) versions 2 and 3, Kerberos, and DNS.

**Active Directory Domain Services (AD DS)**: A directory service (DS) implemented by a [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). The DS provides a data store for [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that is distributed across multiple [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). The [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) interoperate as peers to ensure that a local change to an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) replicates correctly across [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). For more information, see [MS-AUTHSOD] section 1.1.1.5.2 and [MS-ADTS]. For information about product versions, see [MS-ADTS] section 1. See also [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90).

**Active Directory Lightweight Directory Services (AD LDS)**: A directory service (DS) implemented by a [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). The most significant difference between [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) and [**Active Directory Domain Services (AD DS)**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024) is that [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) does not host [**domain naming contexts (domain NCs)**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef). A server can host multiple [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). Each [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) is an independent [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) instance, with its own independent state. [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) can be run as an operating system DS or as a directory service provided by a standalone application (Active Directory Application Mode (ADAM)). For more information, see [MS-ADTS]. See also [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90).

**ancestor object**: An [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) A is an ancestor of [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) O if there is a directed path from A to O (in other words, A is on the path from O to the root of the tree containing O).

**application naming context (application NC)**: A specific type of [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942), or an instance of that type, that supports only [**full replicas**](#gt_f523a137-bda8-45a0-8c9b-f54d86b00bcb) (no partial replicas). An [**application NC**](#gt_53fee475-b07f-45e1-b4b7-c7ac0c1e7f6a) cannot contain security principal objects in Active Directory Domain Services (AD DS), but can contain security principal objects in Active Lightweight Directory Services (AD LDS). A [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) can have zero or more [**application NCs**](#gt_53fee475-b07f-45e1-b4b7-c7ac0c1e7f6a) in either AD DS or AD LDS. An application NC can contain dynamic objects. [**Application NCs**](#gt_53fee475-b07f-45e1-b4b7-c7ac0c1e7f6a) do not appear in the [**global catalog (GC)**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169). The root of an [**application NC**](#gt_53fee475-b07f-45e1-b4b7-c7ac0c1e7f6a) is an object of class [**domainDNS**](#gt_646fb2d2-a783-4a48-832b-bd8491d54f1c).

**attribute**: An identifier for a single or multivalued data element that is associated with a directory [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). An [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) consists of its [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) and their values. For example, cn (common name), street (street address), and mail (email addresses) can all be [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of a [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17). An [**attribute's**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) schema, including the syntax of its values, is defined in an attributeSchema [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

**attribute syntax**: Specifies the format and range of permissible values of an attribute. The syntax of an attribute is defined by several attributes on the attributeSchema object, as specified in [MS-ADTS] section 3.1.1.2. Attribute syntaxes supported by [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) include Boolean, Enumeration, Integer, LargeInteger, String(UTC-Time), Object(DS-DN), and String(Unicode).

**authentication**: The ability of one entity to determine the identity of another entity.

**authentication level**: A numeric value indicating the level of [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) or message protection that [**remote procedure call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) will apply to a specific message exchange. For more information, see [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 13.1.2.1 and [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

**binary large object (BLOB)**: A collection of binary data stored as a single entity in a database.

**binary OID**: An [**object identifier (OID)**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2) in a Basic Encoding Rules (BER)–encoded binary format, as specified in [[ITUX690]](https://go.microsoft.com/fwlink/?LinkId=89924) section 8.19.

**built-in principal**: A [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) within the built-in domain whose [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) is identical in every [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

**canonical name**: A syntactic transformation of an [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**distinguished name (DN)**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) into something resembling a path that still identifies an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) within a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) "cn=Peter Houston, ou=NTDEV, dc=microsoft, dc=com" translates to the canonical name "microsoft.com/NTDEV/Peter Houston", while the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) "dc=microsoft, dc=com" translates to the canonical name "microsoft.com/".

**checksum**: A value that is the summation of a byte stream. By comparing the checksums computed from a data item at two different times, one can quickly assess whether the data items are identical.

**child object, children**: An [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that is not the root of its tree. The children of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) o are the set of all [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) whose parent is o. See section 1 of [MS-ADTS] and section 1 of [MS-DRSR].

**class**: See [**object class**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a).

**compression chunk**: Portions of replication data that occur when compression is used for that data. Compression chunks are created by dividing the replication data into smaller units that are suitable for the particular algorithm. The chunk size is specific to the compression algorithm being employed.

**computer object**: An [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of class computer. A [**computer object**](#gt_d8e8f5a7-db85-40a8-98ed-1abab2237b82) is a [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca); the principal is the operating system running on the computer. The shared secret allows the operating system running on the computer to authenticate itself independently of any user running on the system. See [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409).

**concrete type**: A type used in this specification whose representation must be standardized for interoperability. Specific cases include types in the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definition of an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface, types sent over [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) but whose representation is unknown to [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331), and types stored as byte strings in [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).

**configuration naming context (config NC)**: A specific type of [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942), or an instance of that type, that contains configuration information. In [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90), a single [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625) is shared among all [**domain controllers (DCs)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) in the forest. A [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625) cannot contain security principal objects.

**constructed attribute**: An attribute whose values are computed from normal attributes (for read) and/or have effects on the values of normal attributes (for write).

**container**: An [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the directory that can serve as the parent for other [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). In the absence of schema constraints, all [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) would be [**containers**](#gt_c3143e71-2ada-417e-83f4-3ef10eff2c56). The schema allows only [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of specific classes to be [**containers**](#gt_c3143e71-2ada-417e-83f4-3ef10eff2c56).

**control access right**: An extended access right that can be granted or denied on an [**access control list (ACL)**](#gt_9f92aa05-dd0a-45f2-88d6-89f1fb654395).

**critical object**: A subset of the [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the default [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942), identified by the attribute isCriticalSystemObject having the value TRUE. The [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that are marked in this way are essential for the operation of a [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) hosting the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942).

**crossRef object**: An [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) residing in the partitions container of the [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625) that describes the properties of a [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942), such as its domain naming service name, operational settings, and so on.

**cyclic redundancy check (CRC)**: An algorithm used to produce a [**checksum**](#gt_fa444149-ef93-4512-a278-2e756295630c) (a small, fixed number of bits) against a block of data, such as a packet of network traffic or a block of a computer file. The CRC is a broad class of functions used to detect errors after transmission or storage. A CRC is designed to catch random errors, as opposed to intentional errors. If errors might be introduced by a motivated and intelligent adversary, a cryptographic hash function should be used instead.

**default naming context (default NC)**: When [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) is operating as [**Active Directory Domain Services (AD DS)**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024), the [**default naming context (default NC)**](#gt_adc2c434-9679-419f-8c8b-b8c234921ad3) is the [**domain naming context (domain NC)**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) whose full [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) is hosted by a [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), except when the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) is a [**read-only domain controller (RODC)**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870), in which case the [**default NC**](#gt_adc2c434-9679-419f-8c8b-b8c234921ad3) is a filtered partial NC replica. When operating as [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024), a DC's default NC is the NC of its default NC replica, and the [**default NC**](#gt_adc2c434-9679-419f-8c8b-b8c234921ad3) contains the [**DC's**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) [**computer object**](#gt_d8e8f5a7-db85-40a8-98ed-1abab2237b82). When [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) is operating as [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab), the [**default NC**](#gt_adc2c434-9679-419f-8c8b-b8c234921ad3) is the [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) specified by the msDS-DefaultNamingContext [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). See [**nTDSDSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8).

**deleted-object**: An [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that has been deleted, but remains in storage until a configured amount of time (the deleted-object lifetime) has passed, after which the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) is transformed to a [**recycled-object**](#gt_156b927d-f1ce-4629-993f-18f0cd5e1e12). Unlike a [**recycled-object**](#gt_156b927d-f1ce-4629-993f-18f0cd5e1e12) or a [**tombstone**](#gt_9d8e0963-13fa-4e19-a97f-7ce6bc90d20f), a [**deleted-object**](#gt_d9c9e99f-74f1-483e-bcb1-310e75ff1344) maintains virtually all the state of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) before deletion, and can be undeleted without loss of information. [**Deleted-objects**](#gt_d9c9e99f-74f1-483e-bcb1-310e75ff1344) exist only when the [**Recycle Bin**](#gt_54624800-58f4-45e9-90bf-c9b52dcf98f3) [**optional feature**](#gt_785b66f1-22b3-450f-97aa-a24a39d04d47) is enabled.

**digest**: The fixed-length output string from a one-way hash function that takes a variable-length input string and is probabilistically unique for every different input string. Also, a cryptographic checksum of a data (octet) stream.

**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 object**: An [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca), which is a specialization of the "object" concept that is described in [MS-ADTS] section 1 or [[MS-DRSR]](%5bMS-DRSR%5d.pdf#Section_f977faaa673e4f66b9bf48c640241d47) section 1, Introduction, under Pervasive Concepts. An [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) can be identified by the objectGUID [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of a [**dsname**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a) according to the matching rules defined in [MS-DRSR] section 5.50, DSNAME. The parent-identifying [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) (not exposed as an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f)) is parent. [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) are similar to [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) entries, as defined in [[RFC2251]](https://go.microsoft.com/fwlink/?LinkId=90325); the differences are specified in [MS-ADTS] section 3.1.1.3.1.

**directory service agent (DSA)**: A term from the X.500 [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) specification [[X501]](https://go.microsoft.com/fwlink/?LinkId=98847) that represents a component that maintains and communicates [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) information.

**distinguished name (DN)**: In [**Lightweight Directory Access Protocol (LDAP)**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d), an LDAP Distinguished Name, as described in [RFC2251] section 4.1.3. The DN of an object is the DN of its parent, preceded by the RDN of the object. For example: CN=David Thompson, OU=Users, DC=Microsoft, DC=COM. For definitions of CN and OU, see [[RFC2256]](https://go.microsoft.com/fwlink/?LinkId=91339) sections 5.4 and 5.12, respectively.

**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)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) and host a member list that identifies all members of the domain, as well as optionally hosting the [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) 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**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90), or the server hosting this service. The service hosts the data store for [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) and interoperates with other [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) to ensure that a local change to an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) replicates correctly across all [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). When [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) is operating as [**Active Directory Domain Services (AD DS)**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024), the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) contains [**full NC replicas**](#gt_f523a137-bda8-45a0-8c9b-f54d86b00bcb) of the [**configuration naming context (config NC)**](#gt_54215750-9443-4383-866c-2a95f79f1625), [**schema naming context (schema NC)**](#gt_754c2f7e-1fe4-4d1b-8976-6dad8d13e450), and one of the [**domain NCs**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) in its [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). If the [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024) [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) is a [**global catalog server (GC server)**](#gt_a5a99ce4-e206-42dc-8874-e103934c5b0d), it contains [**partial NC replicas**](#gt_2d142c30-79c2-47f7-81d0-6ae878c5db2c) of the remaining [**domain NCs**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) in its [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). For more information, see [MS-AUTHSOD] section 1.1.1.5.2 and [MS-ADTS]. When [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) is operating as [**Active Directory Lightweight Directory Services (AD LDS)**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab), several [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) can run on one server. When [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) is operating as [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024), only one [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024) [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) can run on one server. However, several [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) can coexist with one [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024) [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) on one server. The [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) contains [**full NC replicas**](#gt_f523a137-bda8-45a0-8c9b-f54d86b00bcb) of the [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625) and the [**schema NC**](#gt_754c2f7e-1fe4-4d1b-8976-6dad8d13e450) in its [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). The domain controller is the server side of Authentication Protocol Domain Support [[MS-APDS]](%5bMS-APDS%5d.pdf#Section_dd444344fd7e430eb3137e95ab9c338e).

**domain name**: A domain name or a NetBIOS name that identifies a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

**domain naming context (domain NC)**: A specific type of [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942), or an instance of that type, that represents a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). A [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) can contain security principal objects; no other type of [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) can contain security principal objects. [**Domain NCs**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) appear in the [**global catalog (GC)**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169). A [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) is hosted by one or more [**domain controllers (DCs)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) operating as [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024). In [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024), a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) has one or more [**domain NCs**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef). A domain NC cannot exist in AD LDS. The root of a [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) is an [**object of class**](#gt_c2c67596-8d8f-42b9-9c70-1c4f7c952200) domainDNS; for directory replication [MS-DRSR], see [**domainDNS**](#gt_646fb2d2-a783-4a48-832b-bd8491d54f1c).

**domain security identifier (domain SID)**: The [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of a [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef). The [**relative identifier (RID)**](#gt_df3d0b61-56cd-4dac-9402-982f1fedc41c) portion of the [**domain SID**](#gt_c1d6ba4d-2302-43a5-acd2-02bfe19d0ade) is always zero. Every security principal object in a [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) has an objectSid [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) equal to the [**domain SID**](#gt_c1d6ba4d-2302-43a5-acd2-02bfe19d0ade) except for the [**RID**](#gt_df3d0b61-56cd-4dac-9402-982f1fedc41c) portion.

**domainDNS**: A specific [**object class**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a). The root of a [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) or an [**application NC**](#gt_53fee475-b07f-45e1-b4b7-c7ac0c1e7f6a) is an [**object of class**](#gt_c2c67596-8d8f-42b9-9c70-1c4f7c952200) domainDNS. The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of such an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) takes the form dc=n1,dc=n2, ... dc=nk, where each ni satisfies the syntactic requirements of a [**fully qualified domain name (FQDN)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) component (for more information, see [[RFC1034]](https://go.microsoft.com/fwlink/?LinkId=90263)). Such a [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) corresponds to the [**FQDN**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) n1.n2. ... .nk. This is the [**FQDN**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) of the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942), and it allows [**replicas**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) of the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) to be located by using DNS.

**DSA GUID**: The objectGUID of a [**DSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8).

**dsname**: A tuple that contains between one and three identifiers for an object. The term [**dsname**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a) does not stand for anything. The possible identifiers are the object's [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) (attribute objectGuid), [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) (attribute objectSid), and [**distinguished name (DN)**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) (attribute distinguishedName). A [**dsname**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a) can appear in a protocol message and as an attribute value (for example, a value of an attribute with syntax Object(DS-DN)). Given a [**DSName**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a), an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) can be identified within a set of [**NC replicas**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) according to the matching rules defined in [MS-DRSR] section 5.49.

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

**dynamic object**: An object with a time-to-die (attribute msDS-Entry-Time-To-Die). The directory service garbage-collects a [**dynamic object**](#gt_ea6b6f3f-6bed-4622-aaca-fd7df28badb9) immediately after its time-to-die has passed. The constructed attribute entryTTL gives a [**dynamic object's**](#gt_ea6b6f3f-6bed-4622-aaca-fd7df28badb9) current time-to-live, that is, the difference between the current time and msDS-Entry-Time-To-Die. For more information, see [[RFC2589]](https://go.microsoft.com/fwlink/?LinkId=90370).

**endpoint**: A network-specific address of a remote procedure call (RPC) server process for remote procedure calls. The actual name and type of the endpoint depends on the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol sequence that is being used. For example, for RPC over TCP (RPC Protocol Sequence ncacn\_ip\_tcp), an endpoint might be TCP port 1025. For RPC over Server Message Block (RPC Protocol Sequence ncacn\_np), an endpoint might be the name of a named pipe. For more information, see [C706].

**expunge**: To permanently remove an object from a [**naming context (NC) replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210), without converting it to a [**tombstone**](#gt_9d8e0963-13fa-4e19-a97f-7ce6bc90d20f).

**extended canonical name**: Same as a [**canonical name**](#gt_79ab9d86-0d30-41c3-b7da-153ad41bdfd8), except that the rightmost forward slash ('/') is replaced with a newline character.

**extended operation**: A special [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16) in which a client [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) requests an action on a [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f); for example, a change in the [**FSMO role owner**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b). FSMO role abandon and FSMO role transfer are examples of extended operations.

**filtered attribute set**: The subset of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that are not replicated to the filtered partial NC replica and the filtered GC partial NC replica. The [**filtered attribute set**](#gt_1bbc9ed8-f11c-4be6-8a41-1f396785602d) is part of the state of the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) and is used to control the [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that replicate to a [**read-only domain controller (RODC)**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870). The searchFlags [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093) [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) is used to define this set.

**flexible single master operation (FSMO)**: A read or [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) operation on a [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942), such that the operation must be performed on the single designated master [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) of that [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942). The master [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) designation is "flexible" because it can be changed without losing the consistency gained from having a single master. This term, pronounced "fizmo", is never used alone; see also [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f), [**FSMO role owner**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b), and FSMO object.

**forest**: For [**Active Directory Domain Services (AD DS)**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024), a set of [**naming contexts (NCs)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) consisting of one [**schema naming context (schema NC)**](#gt_754c2f7e-1fe4-4d1b-8976-6dad8d13e450), one [**configuration naming context (config NC)**](#gt_54215750-9443-4383-866c-2a95f79f1625), one or more [**domain naming contexts (domain NCs)**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef), and zero or more [**application naming contexts (application NCs)**](#gt_53fee475-b07f-45e1-b4b7-c7ac0c1e7f6a). Because a set of [**NCs**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) can be arranged into a tree structure, a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) is also a set containing one or several trees of [**NCs**](#gt_784c7cce-f782-48d8-9444-c9030ba86942). For [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab), a set of [**NCs**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) consisting of one [**schema NC**](#gt_754c2f7e-1fe4-4d1b-8976-6dad8d13e450), one [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625), and zero or more [**application NCs**](#gt_53fee475-b07f-45e1-b4b7-c7ac0c1e7f6a). (In Microsoft documentation, an [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) is called a "configuration set".)

**forest root domain NC**: For [**Active Directory Domain Services (AD DS)**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024), the [**domain naming context (domain NC)**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) within a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) whose child is the [**forest's**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) [**configuration naming context (config NC)**](#gt_54215750-9443-4383-866c-2a95f79f1625). The [**fully qualified domain name (FQDN)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) of the [**forest root domain NC**](#gt_9259fc5d-b976-44b0-b9a8-f7fe5e5ecf85) serves as the [**forest's**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) name.

**forward link attribute**: An [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) whose values include [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) references (for example, an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of syntax Object(DS-DN)). The forward link values can be used to compute the values of a related [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f), a back link attribute, on other [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). If an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) o refers to [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) r in [**forward link attribute**](#gt_ca910b1e-dfb2-4a06-94a8-425013020fb9) f, and there exists a back link attribute b corresponding to f, then a back link value referring to o exists in [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) b on object r. The relationship between the forward and back link attributes is expressed using the linkId [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the attributeSchema [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) representing the two [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f). The forward link's linkId is an even number, and the back link's linkId is the forward link's linkId plus one. A [**forward link attribute**](#gt_ca910b1e-dfb2-4a06-94a8-425013020fb9) can exist with no corresponding back link attribute, but not vice-versa. For more information, see [MS-ADTS].

**FSMO role**: A set of objects that can be updated in only one [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) replica (the FSMO role owner's replica) at any given time. For more information, see [MS-ADTS] section 3.1.1.1.11. See also [**FSMO role owner**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b).

**FSMO role object**: An [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in a [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) that represents a specific [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f). This [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) is an element of the [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f) and contains the fSMORoleOwner attribute.

**FSMO role owner**: The [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) holding the [**naming context (NC) replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) in which the [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of a [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f) can be updated.

**full NC replica**: A [**naming context (NC) replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) that contains all the [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of the [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) it contains. A [**full replica**](#gt_f523a137-bda8-45a0-8c9b-f54d86b00bcb) accepts originating updates.

**fully qualified domain name (FQDN)**: (1) An unambiguous domain name that gives an absolute location in the Domain Name System's (DNS) hierarchy tree, as defined in [[RFC1035]](https://go.microsoft.com/fwlink/?LinkId=90264) section 3.1 and [[RFC2181]](https://go.microsoft.com/fwlink/?LinkId=127732) section 11.

(2) In [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90), a [**fully qualified domain name (FQDN) (1)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) that identifies a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

**GC partial attribute set (PAS)**: The subset of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that replicate to a GC partial NC replica. A particular [**GC partial attribute set (PAS)**](#gt_88536a16-ced1-4fbb-8bf4-8e4d994562af) is part of the state of the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) and is used to control the [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that replicate to [**global catalog servers (GC servers)**](#gt_a5a99ce4-e206-42dc-8874-e103934c5b0d). The isMemberOfPartialAttributeSet [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093) [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) is used to define this set.

**global catalog (GC)**: A unified partial view of multiple [**naming contexts (NCs)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) in a distributed partitioned directory. The [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) directory service [**GC**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169) is implemented by [**GC servers**](#gt_a5a99ce4-e206-42dc-8874-e103934c5b0d). The definition of [**global catalog**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169) is specified in [MS-ADTS] section 3.1.1.1.8.

**global catalog server (GC server)**: A [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that contains a [**naming context (NC) replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) (one full, the rest partial) for each [**domain naming context**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62).

**global group**: An [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) that allows [**user objects**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) from its own [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) and [**global groups**](#gt_2002f42a-84dd-4401-ac8b-8088af87eae6) from its own [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) as members. Also called domain global group. [**Universal groups**](#gt_f46053d6-0708-4094-ac63-57c1bcb73d32) can contain [**global groups**](#gt_2002f42a-84dd-4401-ac8b-8088af87eae6). A [**group object**](#gt_7ce4771c-2043-49b8-85d3-0c60c7789f9a) g is a [**global group**](#gt_2002f42a-84dd-4401-ac8b-8088af87eae6) if and only if GROUP\_TYPE\_ACCOUNT\_GROUP is present in g! groupType; see [MS-ADTS] section 2.2.12, "Group Type Flags". A [**global group**](#gt_2002f42a-84dd-4401-ac8b-8088af87eae6) that is also a security-enabled group is valid for inclusion within ACLs anywhere in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). If a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) is in mixed mode, then a [**global group**](#gt_2002f42a-84dd-4401-ac8b-8088af87eae6) in that [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) that is also a security-enabled group allows only [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) as members. See also domain local group, security-enabled group.

**globally unique identifier (GUID)**: A term used interchangeably with [**universally unique identifier (UUID)**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) 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]](https://go.microsoft.com/fwlink/?LinkId=90460) or [C706] must be used for generating the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1). See also [**universally unique identifier (UUID)**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3).

**group**: A [**group object**](#gt_7ce4771c-2043-49b8-85d3-0c60c7789f9a).

**group object**: In [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90), a [**group object**](#gt_7ce4771c-2043-49b8-85d3-0c60c7789f9a) has an [**object class**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a) group. A [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) has a [**forward link attribute**](#gt_ca910b1e-dfb2-4a06-94a8-425013020fb9) member; the values of this [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) either represent elements of the [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) (for example, [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of class user or computer) or subsets of the [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) ([**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of class group). The representation of group subsets is called "nested group membership". The back link attribute memberOf enables navigation from [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) members to the [**groups**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) containing them. Some [**groups**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) represent [**groups**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) of [**security principals**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) and some do not and are, for instance, used to represent email distribution lists.

**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.

**Internet host name**: The name of a host as defined in [[RFC1123]](https://go.microsoft.com/fwlink/?LinkId=90268) section 2.1, with the extensions described in [[MS-HNDS]](%5bMS-HNDS%5d.pdf#Section_eff5b201ad32485dbbed1d07ad069d5c).

**invocation ID**: The invocationId [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f). An [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of an [**nTDSDSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8). Its value is a unique identifier for a function that maps from [**update sequence numbers (USNs)**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) to [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) to the [**NC replicas**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) of a [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). See also [**nTDSDSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8).

**Knowledge Consistency Checker (KCC)**: A component of the [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) replication that is used to create spanning trees for [**domain controller**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) to [**domain controller**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) replication and to translate those trees into settings of variables that implement the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) topology.

**LDAP connection**: A TCP connection from a client to a server over which the client sends Lightweight Directory Access Protocol (LDAP) requests and the server sends responses to the client's requests.

**Lightweight Directory Access Protocol (LDAP)**: The primary access protocol for [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90). Lightweight Directory Access Protocol (LDAP) is an industry-standard protocol, established by the Internet Engineering Task Force (IETF), which allows users to query and update information in a directory service (DS), as described in [MS-ADTS]. The Lightweight Directory Access Protocol can be either version 2 [[RFC1777]](https://go.microsoft.com/fwlink/?LinkId=90290) or version 3 [[RFC3377]](https://go.microsoft.com/fwlink/?LinkID=91337).

**lingering object**: An [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that still exists in an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) even though it has been deleted and garbage-collected from other [**replicas**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac). This occurs, for instance, when a [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) goes offline for longer than the tombstone lifetime.

**link attribute**: A [**forward link attribute**](#gt_ca910b1e-dfb2-4a06-94a8-425013020fb9) or a back link attribute.

**link value**: The value of a [**link attribute**](#gt_be41074d-ce6b-4488-853a-4bbb3ea243ce).

**link value stamp**: The type of a [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) attached to a [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238).

**local domain controller (local DC)**: A [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) on which the current method is executing.

**Lost and Found container**: A container holding objects in a given [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) that do not have parent objects due to add and remove operations that originated on different [**domain controllers (DCs)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). The container is a child of the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root and has RDN CN=LostAndFound in [**domain NCs**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) and CN=LostAndFoundConfig in [**config NCs**](#gt_54215750-9443-4383-866c-2a95f79f1625).

**Microsoft Interface Definition Language (MIDL)**: The Microsoft implementation and extension of the OSF-DCE [**Interface Definition Language (IDL)**](#gt_73177eec-4092-420f-92c5-60b2478df824). [**MIDL**](#gt_9c5903c1-1477-4181-b451-3ba1e34a0c0c) can also mean the [**Interface Definition Language (IDL)**](#gt_73177eec-4092-420f-92c5-60b2478df824) compiler provided by Microsoft. For more information, see [MS-RPCE].

**MSZIP compression algorithm**: A compression algorithm specified in [[RFC1951]](https://go.microsoft.com/fwlink/?LinkId=90302) that is used between Windows 2000 operating system [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**naming context (NC)**: An [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) is a set of objects organized as a tree. It is referenced by a DSName. The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the DSName is the distinguishedName [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of the tree root. The [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) of the DSName is the objectGUID [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of the tree root. The [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the DSName, if present, is the objectSid [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of the tree root; for [**Active Directory Domain Services (AD DS)**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024), the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) is present if and only if the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) is a [**domain naming context (domain NC)**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef). [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) supports organizing several [**NCs**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) into a tree structure.

**NC replica**: A variable containing a tree of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) whose root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) is identified by some [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942).

**NetBIOS domain name**: The name registered by [**domain controllers (DCs)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) on [1C] records of the NBNS (WINS) server (see section 6.3.4). For details of NetBIOS name registration, see [[MS-WPO]](%5bMS-WPO%5d.pdf#Section_c5f54a7765be40a0bb829e4181d8ab67) sections 7.1.4 and 10.4.

**nonreplicated attribute**: An attribute whose values are not replicated between [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) replicas. The nonreplicated attributes of an object are, in effect, local variables of the [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) hosting the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) replica containing that object, since changes to these attributes have no effect outside that [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**nTDSDSA object**: An [**object of class**](#gt_c2c67596-8d8f-42b9-9c70-1c4f7c952200) nTDSDSA that is always located in the [**configuration naming context (config NC)**](#gt_54215750-9443-4383-866c-2a95f79f1625). This [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) represents a [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). See [MS-ADTS] section 6.1.1.2.2.1.2.1.1.

**NULL GUID**: A [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) of all zeros.

**object**: A set of attributes, each with its associated values. For more information on objects, see [MS-ADTS] section 1 or [MS-DRSR] section 1.

**object class**: A set of restrictions on the construction and update of objects. An [**object class**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a) can specify a set of must-have attributes (every object of the class must have at least one value of each) and may-have attributes (every object of the class may have a value of each). An [**object class**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a) can also specify the allowable classes for the parent object of an object in the class. An [**object class**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a) can be defined by single inheritance; an object whose class is defined in this way is a member of all [**object classes**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a) used to derive its most specific class. An [**object class**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a) is defined in a classSchema object. See section 1 of [MS-ADTS] and section 1 of [MS-DRSR].

**object identifier (OID)**: In the Lightweight Directory Access Protocol (LDAP), a sequence of numbers in a format described by [[RFC1778]](https://go.microsoft.com/fwlink/?LinkId=90291). In many LDAP directory implementations, an OID is the standard internal representation of an attribute. In the directory model used in this specification, the more familiar ldapDisplayName represents an attribute.

**object of class x (or x object)**: An object o such that one of the values of its objectClass attributes is x. For instance, if objectClass contains the value user, o is an object of class user. This is often contracted to "user object".

**object reference**: An [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) value that references an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). Reading a reference gives the [**distinguished name (DN)**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

**objectClass**: The objectClass [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f). The [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that holds the object class name of each [**object class**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

**objectGUID**: The [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on an [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) object whose value is a [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) that uniquely identifies the object. The [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) value of an [**object's**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) objectGUID is assigned when the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) was created and is immutable thereafter. The integrity of [**object references**](#gt_3ca938ae-c14f-4f59-8a7d-daca9f76db4e) between [**NCs**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) and of [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) depends on the integrity of the objectGUID [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f). For a descrption of the general concept of an "object", see [MS-ADTS] section 1. For more detailed information see [MS-ADTS] section 3.1.1.1.3.

**objectSid**: The objectSid [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f). The [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) whose value is a [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) that identifies the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) as a security principal object. The value of an [**object's**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) objectSid is assigned when the security principal object was created and is immutable thereafter unless the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) moves to another [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). The integrity of [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) depends on the integrity of the objectSid [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).

**opnum**: An operation number or numeric identifier that is used to identify a specific [**remote procedure call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) method or a method in an interface. For more information, see [C706] section 12.5.2.12 or [MS-RPCE].

**optional feature**: A non-default behavior that modifies the [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) state model. An [**optional feature**](#gt_785b66f1-22b3-450f-97aa-a24a39d04d47) is enabled or disabled in a specific scope, such as a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) or a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). For more information, refer to [MS-ADTS] section 3.1.1.9.

**oriented tree**: A directed acyclic graph such that for every vertex v, except one (the root), there is a unique edge whose tail is v. There is no edge whose tail is the root. For more information, see [KNUTH1] section 2.3.4.2.

**originating update**: An update that is performed to an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) via any protocol except replication. An [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) to an attribute or link value generates a new [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) for the attribute or link value.

**parent object**: An object is either the root of a tree of objects or has a parent. If two objects have the same parent, they must have different values in their [**relative distinguished names (RDNs)**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9). See also, [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in section 1 of [MS-ADTS] and section 1 of [MS-DRSR].

**partial attribute set (PAS)**: The subset of attributes that replicate to partial [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) replicas. Also, the particular partial attribute set that is part of the state of a forest and that is used to control the attributes that replicate to [**global catalog (GC)**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169) servers.

**partial NC replica**: An [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) that contains a [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093)-specified subset of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) for the [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) it contains. A [**partial NC replica**](#gt_2d142c30-79c2-47f7-81d0-6ae878c5db2c) is not writable—it does not accept [**originating updates**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20). See also [**writable NC replica**](#gt_51db485c-dcf6-4845-99b3-2df414ef0aa9).

**Partitions container**: A [**child object**](#gt_9b04b599-9dca-48f1-aa9e-08e254d20553) of the [**configuration naming context (config NC)**](#gt_54215750-9443-4383-866c-2a95f79f1625) root. The [**relative distinguished name (RDN)**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) of the [**Partitions container**](#gt_d6b4c198-f9d3-4c49-b0f0-390e07f89af1) is "cn=Partitions" and its class is crossRefContainer ([MS-ADTS] section 2.30). See also [**crossRef object**](#gt_353fac65-0774-4ba8-8081-eb4c963f94e7).

**PDC emulator**: A [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that is designated to track changes made to the accounts of all computers in a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). The [**PDC emulator**](#gt_48b8ecd1-32ae-4593-88e6-346ece75ef34) 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**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) database. A [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) has only one [**PDC emulator**](#gt_48b8ecd1-32ae-4593-88e6-346ece75ef34).

**prefix table**: A data structure that is used to translate between an [**object identifier (OID)**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2) and a compressed representation for [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2). See [MS-DRSR] section 5.14.

**primary domain controller (PDC)**: A [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) designated to track changes made to the accounts of all computers on a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). 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**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) database. A [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) has only one [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d).

**primary domain controller (PDC) role owner**: The [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that hosts the [**primary domain controller**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) emulator FSMO role for a given domain [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942).

**principal**: See [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409).

**Privileged Access Management**: An optional feature that enables the removal of a link value from the state of a [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) at a specified date and time.

**read permission**: The authorization to read an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). For more information, see [MS-ADTS] section 5.1.3.

**read-only domain controller (RODC)**: A [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that does not accept [**originating updates**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20). Additionally, an [**RODC**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870) does not perform outbound [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb). An RODC cannot be the primary domain controller (PDC) for its domain.

**Recycle Bin**: An [**optional feature**](#gt_785b66f1-22b3-450f-97aa-a24a39d04d47) that modifies the state model of object deletions and undeletions, making undeletion of [**deleted-objects**](#gt_d9c9e99f-74f1-483e-bcb1-310e75ff1344) possible without loss of the object's attribute values. For more information, see [MS-ADTS] section 3.1.1.9.1.

**recycled-object**: An [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that has been deleted, but remains in storage until a configured amount of time (the tombstone lifetime) has passed, after which the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) is permanently removed from storage. Unlike a [**deleted-object**](#gt_d9c9e99f-74f1-483e-bcb1-310e75ff1344), most of the state of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) has been removed, and the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) can no longer be undeleted without loss of information. By keeping the [**recycled-object**](#gt_156b927d-f1ce-4629-993f-18f0cd5e1e12) in existence for the tombstone lifetime, the deleted state of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) is able to replicate. [**Recycled-objects**](#gt_156b927d-f1ce-4629-993f-18f0cd5e1e12) exist only when the [**Recycle Bin**](#gt_54624800-58f4-45e9-90bf-c9b52dcf98f3) [**optional feature**](#gt_785b66f1-22b3-450f-97aa-a24a39d04d47) is enabled.

**relative distinguished name (RDN)**: The name of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) relative to its parent. This is the leftmost attribute-value pair in the [**distinguished name (DN)**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). For example, in the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) "cn=Peter Houston, ou=NTDEV, dc=microsoft, dc=com", the [**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) is "cn=Peter Houston". For more information, see [RFC2251].

**relative identifier (RID)**: The last item in the series of SubAuthority values in a [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) [[SIDD]](https://go.microsoft.com/fwlink/?LinkId=90516). It distinguishes one account or group from all other accounts and groups in the domain. No two accounts or groups in any domain share the same RID.

**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].

**replica**: A variable containing a set of objects.

**replicated attribute**: An [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) whose values are replicated to other [**NC replicas**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). An [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) is replicated if its attributeSchema [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) o does not have a value for the systemFlags [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f), or if the FLAG\_ATTR\_NOT\_REPLICATED bit (bit 0) of o! systemFlags is zero.

**replicated update**: An update performed to a [**naming context (NC) replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) by the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) system, to propagate the effect of an [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) at another [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). The [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) assigned during the [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) to attribute values or a link value is preserved by [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb).

**replication**: The process of propagating the effects of all originating writes to any replica of a [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942), to all replicas of the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942). If originating writes cease and replication continues, all replicas converge to a common application-visible state.

**replication cycle**: Sometimes referred to simply as "cycle". A series of one or more [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) responses associated with the same [**invocationId**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a), concluding with the return of a new up-to-date vector.

**replication epoch**: A state variable of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that changes when a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) is no longer compatible for [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) with its former partners. A server receiving a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) request tests the client's [**replication epoch**](#gt_cb4c7bb2-7c28-4ce0-b5f6-de93a7e236d8) against its own, and refuses the request if the two are not equal.

**replication latency**: The time lag between a final originating update to a [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) replica and all [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) replicas reaching a common application-visible state.

**RPC protocol sequence**: A character string that represents a valid combination of a [**remote procedure call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol, a network layer protocol, and a transport layer protocol, as described in [C706] and [MS-RPCE].

**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.

**schema**: The set of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) and [**object classes**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a) that govern the creation and update of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

**schema naming context (schema NC)**: A specific type of [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) or an instance of that type. A [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) has a single [**schema NC**](#gt_754c2f7e-1fe4-4d1b-8976-6dad8d13e450), which is replicated to each [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). No other [**NC replicas**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) can contain these [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). Each [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) and class in the [**forest's**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) schema is represented as a corresponding [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the [**forest's**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) [**schema NC**](#gt_754c2f7e-1fe4-4d1b-8976-6dad8d13e450). A schema NC cannot contain security principal objects.

**secret data**: An implementation-specific set of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on [**objects of class**](#gt_c2c67596-8d8f-42b9-9c70-1c4f7c952200) user that contain security-sensitive information about the [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409).

**security context**: A data structure containing authorization information for a particular [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) in the form of a collection of [**security identifiers (SIDs)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d). One [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) identifies the [**principal**](#gt_8492780e-99e2-47ba-8553-aedb8de9f9c0) specifically, whereas others represent other capabilities. A server uses the authorization information in a [**security context**](#gt_88d49f20-6c95-4b64-a52c-c3eca2fe5709) to check access to requested resources.

**security descriptor**: A data structure containing the security information associated with a securable [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). A [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) identifies an [**object's**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) owner by its [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d). If access control is configured for the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca), its [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) contains a discretionary access control list (DACL) with [**SIDs**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) for the [**security principals**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) who are allowed or denied access. Applications use this structure to set and query an [**object's**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) security status. The [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) is used to guard access to an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) as well as to control which type of auditing takes place when the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) is accessed. The [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) format is specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.6; a string representation of [**security descriptors**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350), called SDDL, is specified in [MS-DTYP] section 2.5.1.

**security identifier (SID)**: An identifier for [**security principals**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) that is used to identify an account or a group. Conceptually, the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) is composed of an account authority portion (typically a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca)) and a smaller integer representing an identity relative to the account authority, termed the [**relative identifier (RID)**](#gt_df3d0b61-56cd-4dac-9402-982f1fedc41c). The [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) format is specified in [MS-DTYP] section 2.4.2; a string representation of [**SIDs**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) is specified in [MS-DTYP] section 2.4.2 and [[MS-AZOD]](%5bMS-AZOD%5d.pdf#Section_5a0a0a3ec7a742e1b5f2cc8d8bd9739e) section 1.1.1.2.

**security principal**: A unique entity, also referred to as a principal, that can be authenticated by [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90). It frequently corresponds to a human user, but also can be a service that offers a resource to other security principals. Other security principals might be a group, which is a set of principals. Groups are supported by [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90).

**security provider**: A pluggable security module that is specified by the protocol layer above the [**remote procedure call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) layer, and will cause the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) layer to use this module to secure messages in a communication session with the server. The security provider is sometimes referred to as an [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) service. For more information, see [C706] and [MS-RPCE].

**server object**: A class of [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the [**configuration naming context (config NC)**](#gt_54215750-9443-4383-866c-2a95f79f1625). A [**server object**](#gt_62a8c543-5998-480b-8fa7-41a8f04a18e5) can have an [**nTDSDSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8) as a child.

**service account**: A stored set of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that represent a [**principal**](#gt_8492780e-99e2-47ba-8553-aedb8de9f9c0) that provides a [**security context**](#gt_88d49f20-6c95-4b64-a52c-c3eca2fe5709) for services.

**service class**: The first part of a [**service principal name**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4). See [[MS-KILE]](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9) section 3.1.5.11.

**service principal name (SPN)**: The name a client uses to identify a service for mutual [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317). (For more information, see [[RFC1964]](https://go.microsoft.com/fwlink/?LinkId=90304) section 2.1.1.) An [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) consists of either two parts or three parts, each separated by a forward slash ('/'). The first part is the [**service class**](#gt_c1386afd-9d42-47c7-a7ea-d01912a17647), the second part is the host name, and the third part (if present) is the service name. For example, "ldap/dc-01.fabrikam.com/fabrikam.com" is a three-part [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) where "ldap" is the service class name, "dc-01.fabrikam.com" is the host name, and "fabrikam.com" is the service name. See [[SPNNAMES]](https://go.microsoft.com/fwlink/?LinkId=90532) for more information about [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) format and composing a unique [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4).

**session key**: A relatively short-lived symmetric key (a cryptographic key negotiated by the client and the server based on a shared secret). A [**session key's**](#gt_4f67a585-fb00-4166-93e8-cf4abca8226d) lifespan is bounded by the session to which it is associated. A [**session key**](#gt_4f67a585-fb00-4166-93e8-cf4abca8226d) has to be strong enough to withstand cryptanalysis for the lifespan of the session.

**SHA1 hash**: A hashing algorithm defined in [[FIPS180]](https://go.microsoft.com/fwlink/?LinkId=89867) that was developed by the National Institute of Standards and Technology (NIST) and the National Security Agency (NSA).

**site**: A collection of one or more well-connected (reliable and fast) TCP/IP subnets. By defining [**sites**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba) (represented by [**site objects**](#gt_0ce6abc5-9823-4a69-bb30-12e42ff99629)) an administrator can optimize both [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) access and [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) with respect to the physical network. When users log in, [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) clients find [**domain controllers (DCs)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that are in the same [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba) as the user, or near the same [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba) if there is no [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) in the [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba). See also [**Knowledge Consistency Checker (KCC)**](#gt_c7d4f1f6-5285-4168-b21a-022f775a3f58). For more information, see [MS-ADTS].

**site object**: An [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of class site, representing a [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba).

**stamp**: Information that describes an [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) by a [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). The [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) is not the new data value; the [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) is information about the update that created the new data value. A [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) is often called metadata, because it is additional information that "talks about" the conventional data values. A [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) contains the following pieces of information: the unique identifier of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that made the [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20); a sequence number characterizing the order of this change relative to other changes made at the originating [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd); a version number identifying the number of times the data value has been modified; and the time when the change occurred.

**STATUS code**: A 32-bit quantity where zero represents success and nonzero represents failure. The specific failure codes used in this specification are [**Windows error codes**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**subordinate reference object (sub-ref object)**: The [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of a parent NC has a list of all the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) roots of its child NCs in the subRefs [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) ([[MS-ADA3]](%5bMS-ADA3%5d.pdf#Section_4517e8353ee644d4bb95a94b6966bfb0) section 2.282). Each entry in this list is a subordinate reference and the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) named by the entry is denominated a [**subordinate reference object**](#gt_a4b4bece-8452-402c-99c6-12ebf0af0b58). An [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) is a [**subordinate reference object**](#gt_a4b4bece-8452-402c-99c6-12ebf0af0b58) if and only if it is in such a list. If a server has [**replicas**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) of both an [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) and its child NC, then the child NC root is the [**subordinate reference object**](#gt_a4b4bece-8452-402c-99c6-12ebf0af0b58), in the context of the parent NC. If the server does not have a [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) of the child NC, then another [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca), with distinguishedName ([[MS-ADA1]](%5bMS-ADA1%5d.pdf#Section_19528560f41e4623a406dabcfff0660f) section 2.177) and objectGUID ([MS-ADA3] section 2.44) [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) equal to the child NC root, is present in the server and is the [**subordinate reference object**](#gt_a4b4bece-8452-402c-99c6-12ebf0af0b58).

**target object**: An object referenced by a forward link value.

**tombstone**: An object that has been deleted, but remains in storage until a configured amount of time (the tombstone lifetime) has passed, after which the object is permanently removed from storage. By keeping the [**tombstone**](#gt_9d8e0963-13fa-4e19-a97f-7ce6bc90d20f) in existence for the tombstone lifetime, the deleted state of the object is able to replicate. [**Tombstones**](#gt_9d8e0963-13fa-4e19-a97f-7ce6bc90d20f) exist only when the [**Recycle Bin**](#gt_54624800-58f4-45e9-90bf-c9b52dcf98f3) [**optional feature**](#gt_785b66f1-22b3-450f-97aa-a24a39d04d47) is not enabled.

**Unicode**: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) standard [[UNICODE5.0.0/2007]](https://go.microsoft.com/fwlink/?LinkId=154659) provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

**universal group**: An [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) that allows [**user objects**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17), [**global groups**](#gt_2002f42a-84dd-4401-ac8b-8088af87eae6), and [**universal groups**](#gt_f46053d6-0708-4094-ac63-57c1bcb73d32) from anywhere in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) as members. A [**group object**](#gt_7ce4771c-2043-49b8-85d3-0c60c7789f9a) g is a [**universal group**](#gt_f46053d6-0708-4094-ac63-57c1bcb73d32) if and only if GROUP\_TYPE\_UNIVERSAL\_GROUP is present in g! groupType. A security-enabled universal group is valid for inclusion within ACLs anywhere in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). If a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) is in mixed mode, then a [**universal group**](#gt_f46053d6-0708-4094-ac63-57c1bcb73d32) cannot be created in that [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). See also domain local group, security-enabled group.

**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**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) objects. UUIDs are highly likely to be unique. UUIDs are also known as [**globally unique identifiers (GUIDs)**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) 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.

**update**: An add, modify, or delete of one or more objects or attribute values. See [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20), [**replicated update**](#gt_2a923099-db0a-4932-af28-4354601e85c4).

**update sequence number (USN)**: A monotonically increasing sequence number used in assigning a stamp to an originating update. For more information, see [MS-ADTS].

**up-to-date vector**: The representation of an assertion about the presence of [**originating updates**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) from different sources in an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). See [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16) and [**update sequence number (USN)**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c).

**user object**: An object of class user. A user object is a security principal object; the principal is a person or service entity running on the computer. The shared secret allows the person or service entity to authenticate itself, as described in ([MS-AUTHSOD] section 1.1.1.1).

**Windows error code**: A 32-bit quantity where zero represents success and nonzero represents failure. The specific failure codes are specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90).

**writable naming context (NC) replica**: A [**naming context (NC)**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) replica that accepts originating updates. A [**writable NC replica**](#gt_51db485c-dcf6-4845-99b3-2df414ef0aa9) is always full, but a [**full NC replica**](#gt_f523a137-bda8-45a0-8c9b-f54d86b00bcb) is not always writable. Partial replicas are not writable. See also read-only full NC replica.

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used as defined in [[RFC2119]](https://go.microsoft.com/fwlink/?LinkId=90317). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

## 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](https://go.microsoft.com/fwlink/?linkid=850906).

### 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](mailto: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](https://go.microsoft.com/fwlink/?LinkId=89824)

[ISO/IEC 13239] International Organization for Standardization, "Information technology -- Telecommunications and information exchange between systems -- High-level data link control (HDLC) procedures", [http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=37010&IC S1=35&ICS2=100&ICS3=20&showrevision=y&scopelist=CATALOGUE)](https://go.microsoft.com/fwlink/?LinkId=98149)

[ITUX690] ITU-T, "ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", Recommendation X.690, July 2002, [http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf](https://go.microsoft.com/fwlink/?LinkId=89924)

[MS-ADA1] Microsoft Corporation, "[Active Directory Schema Attributes A-L](%5bMS-ADA1%5d.pdf#Section_19528560f41e4623a406dabcfff0660f)".

[MS-ADA2] Microsoft Corporation, "[Active Directory Schema Attributes M](%5bMS-ADA2%5d.pdf#Section_e20ebc4e528540bab3bdffcb81c2783e)".

[MS-ADA3] Microsoft Corporation, "[Active Directory Schema Attributes N-Z](%5bMS-ADA3%5d.pdf#Section_4517e8353ee644d4bb95a94b6966bfb0)".

[MS-ADLS] Microsoft Corporation, "[Active Directory Lightweight Directory Services Schema](%5bMS-ADLS%5d.pdf#Section_9427994325ab4c139bf26d411cc2f796)".

[MS-ADSC] Microsoft Corporation, "[Active Directory Schema Classes](%5bMS-ADSC%5d.pdf#Section_9abb5e97123d4da99557b353ab79b830)".

[MS-ADTS] Microsoft Corporation, "[Active Directory Technical Specification](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a)".

[MS-DTYP] Microsoft Corporation, "[Windows Data Types](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2)".

[MS-ERREF] Microsoft Corporation, "[Windows Error Codes](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90)".

[MS-KILE] Microsoft Corporation, "[Kerberos Protocol Extensions](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9)".

[MS-LSAD] Microsoft Corporation, "[Local Security Authority (Domain Policy) Remote Protocol](%5bMS-LSAD%5d.pdf#Section_1b5471ef4c334a91b079dfcbb82f05cc)".

[MS-NRPC] Microsoft Corporation, "[Netlogon Remote Protocol](%5bMS-NRPC%5d.pdf#Section_ff8f970f3e3740f7bd4baf7336e4792f)".

[MS-RPCE] Microsoft Corporation, "[Remote Procedure Call Protocol Extensions](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)".

[MS-SAMR] Microsoft Corporation, "[Security Account Manager (SAM) Remote Protocol (Client-to-Server)](%5bMS-SAMR%5d.pdf#Section_4df07fab1bbc452f8e927853a3c7e380)".

[MS-SRPL] Microsoft Corporation, "[Directory Replication Service (DRS) Protocol Extensions for SMTP](%5bMS-SRPL%5d.pdf#Section_ec69eea50d5e428ab5bc66732aaeb866)".

[RC4] RSA Data Security, Inc., "The RC4 Encryption Algorithm", [http://www.rsa.com/node.aspx?id=1204](https://go.microsoft.com/fwlink/?LinkId=93759)

[RFC1034] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987, [http://www.ietf.org/rfc/rfc1034.txt](https://go.microsoft.com/fwlink/?LinkId=90263)

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992, [http://www.ietf.org/rfc/rfc1321.txt](https://go.microsoft.com/fwlink/?LinkId=90275)

[RFC1951] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996, [http://www.ietf.org/rfc/rfc1951.txt](https://go.microsoft.com/fwlink/?LinkId=90302)

[RFC1964] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, June 1996, [http://www.rfc-editor.org/rfc/rfc1964.txt](https://go.microsoft.com/fwlink/?LinkId=90304)

[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](https://go.microsoft.com/fwlink/?LinkId=90317)

[RFC2251] Wahl, M., Howes, T., and Kille, S., "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997, [http://www.ietf.org/rfc/rfc2251.txt](https://go.microsoft.com/fwlink/?LinkId=90325)

[RFC2252] Wahl, M., Coulbeck, A., Howes, T., and Kille, S., "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions", RFC 2252, December 1997, [http://www.ietf.org/rfc/rfc2252.txt](https://go.microsoft.com/fwlink/?LinkId=90326)

[RFC2253] Wahl, M., Kille, S., and Howe, T., "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", RFC 2253, December 1997, [http://www.ietf.org/rfc/rfc2253.txt](https://go.microsoft.com/fwlink/?LinkId=90327)

[RFC2254] Howes, T., "The String Representation of LDAP Search Filters", RFC 2254, December 1997, [http://www.ietf.org/rfc/rfc2254.txt](https://go.microsoft.com/fwlink/?LinkId=90328)

[RFC2821] Klensin, J., "Simple Mail Transfer Protocol", RFC 2821, April 2001, [http://www.ietf.org/rfc/rfc2821.txt](https://go.microsoft.com/fwlink/?LinkId=90384)

[RFC4122] Leach, P., Mealling, M., and Salz, R., "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005, [http://www.rfc-editor.org/rfc/rfc4122.txt](https://go.microsoft.com/fwlink/?LinkId=90460)

### Informative References

[KNUTH1] Knuth, D., "The Art of Computer Programming: Volume 1/Fundamental Algorithms (Second Edition)", Reading, MA: Addison-Wesley, 1973, ASIN: B000NV8YOA.

[MS-ADOD] Microsoft Corporation, "[Active Directory Protocols Overview](%5bMS-ADOD%5d.pdf#Section_5ff67bf4c14548cb89cd4f5482d94664)".

[MS-LSAT] Microsoft Corporation, "[Local Security Authority (Translation Methods) Remote Protocol](%5bMS-LSAT%5d.pdf#Section_1ba21e6fd8a9462c91534375f2020894)".

[NELSON] Nelson, G., "Systems Programming with Modula-3", Englewood Cliffs, NJ: Prentice Hall, 1991, ISBN: 0135904641.

## Overview

This document specifies the Directory Replication Service (DRS) Remote Protocol, an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol for [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) between [**domain controllers**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) and management of [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90). The protocol consists of two RPC interfaces, named drsuapi and dsaop. The name of each drsuapi method begins with "IDL\_DRS", while the name of each dsaop method begins with "IDL\_DSA".

### Methods Categorized by Function

The DRS Remote Protocol contains methods that are diverse in function and fall into the following categories:

* Context handle methods: [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d), [IDL\_DRSUnbind](#Section_49eb17c9b6a94ceabef866abda8a7850). These methods create and destroy [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handles that maintain volatile state used by drsuapi methods. The dsaop methods do not use context handles.
* Replication methods: IDL\_DRSGetNCChanges, IDL\_DRSReplicaSync, IDL\_DRSReplicaVerifyObjects, IDL\_DRSGetReplInfo. The IDL\_DRSGetNCChanges method replicates [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) changes from the server to the client. The IDL\_DRSReplicaSync and IDL\_DRSReplicaVerifyObjects methods cause the server to call IDL\_DRSGetNCChanges on the client. The IDL\_DRSGetReplInfo method is used to gather information about the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state of the server.
* Cross-domain move method: IDL\_DRSInterDomainMove. This method is used in the server implementation of [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) Modify [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) when the DN modification moves an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) from one [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) to another.
* Lookups: IDL\_DRSVerifyNames, IDL\_DRSCrackNames, IDL\_DRSGetMemberships, IDL\_DRSGetMemberships2. These methods perform specialized directory lookups. They are all used by a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) client; the IDL\_DRSCrackNames method is commonly used by a non-DC client.
* DC Locator support methods: IDL\_DRSDomainControllerInfo, IDL\_DRSQuerySitesByCost. These methods retrieve information about the domain controllers in a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) or [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) and information about the cost of connections between different [**sites**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba).
* Windows NT 4.0 Replication support method: IDL\_DRSGetNT4ChangeLog. This method is used in the implementation of [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) support for replication to Windows NT 4.0 backup domain controllers (BDCs), specifically in the implementation of moving the [**PDC Emulator**](#gt_48b8ecd1-32ae-4593-88e6-346ece75ef34) [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f) from one DC to another without triggering a full sync of Windows NT 4.0 BDCs (see [[MS-NRPC]](%5bMS-NRPC%5d.pdf#Section_ff8f970f3e3740f7bd4baf7336e4792f) section 3.6).
* [**Knowledge Consistency Checker (KCC)**](#gt_c7d4f1f6-5285-4168-b21a-022f775a3f58) support methods: IDL\_DRSUpdateRefs, IDL\_DRSReplicaAdd, IDL\_DRSReplicaDel, IDL\_DRSReplicaModify, IDL\_DRSExecuteKCC. These methods are used by the KCC ([[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.2) and by administrator tools to manage replication topology.
* Administrator-tool support methods: IDL\_DRSAddEntry, IDL\_DRSAddSidHistory, IDL\_DRSRemoveDsServer, IDL\_DRSRemoveDsDomain, IDL\_DRSGetObjectExistence, IDL\_DSAPrepareScript, IDL\_DSAExecuteScript, IDL\_DRSWriteSPN, IDL\_DRSInitDemotion, IDL\_DRSFinishDemotion, IDL\_DRSReplicaDemotion, IDL\_DRSAddCloneDC. These methods are used by administrator tools to perform various specialized functions.
* msDS-KeyCredentialLink attribute support methods: [IDL\_DRSWriteNgcKey](#Section_7a140389caa34718bb1ad64483933eb0), [IDL\_DRSReadNgcKey](#Section_a80c60ac9864444a95136c0c894fbb8d). These methods are used to create or query the msDS-KeyCredentialLink attribute on a computer account.

The specification of each method in section [4](#Section_9554afa5e7554742a34b899fc4e2fd20), includes an *Informative summary of behavior* that provides a detailed introduction to the method.

### Sequencing Issues

The sequencing issues in this [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol are as follows:

* For server and client initialization, see section [3.6](#Section_37b8ea496c8b45f6aa1b1125df02f0e4).
* The drsuapi RPC interface is a "context handle"–based RPC interface; [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) specifies what this means. A client obtains a [DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) for a particular [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) by calling [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d), then calls any other drsuapi method on that DC, passing the DRS\_HANDLE as the first parameter. The client's DRS\_HANDLE remains valid for making method calls until the client calls [IDL\_DRSUnbind](#Section_49eb17c9b6a94ceabef866abda8a7850), or until the server unilaterally invalidates the DRS\_HANDLE (for example, by crashing).

The only state associated with a DRS\_HANDLE is the state established by IDL\_DRSBind. This state is immutable for as long as the DRS\_HANDLE remains valid. Therefore, if a client creates two binding handles to the same DC by using the same parameters to IDL\_DRSBind, the server behavior of a drsuapi method is not affected by the client's choice of binding handle passed to the method.

Because the state associated with a DRS\_HANDLE is immutable so long as the DRS\_HANDLE remains valid, there are no special considerations involved in making concurrent method calls using the same DRS\_HANDLE; the client is free to make concurrent method calls.

* Two methods use the "cookie" design pattern. In this pattern, the client sends an initial request containing a designated null value for a certain parameter. The server response contains a value that is opaque to the client or contains the designated null value. If the value is not null, and the response indicates that another client request is required to complete some higher-level operation, the client sends the opaque value returned by the server in the next request rather than sending the designated null value. The exchange of requests and responses continues until some response indicates that the higher-level operation is complete.

The two methods that follow this pattern are:

* + [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894): In this instance of the "cookie" pattern, the server returns a "cookie" in the response that completes the higher-level operation. The client can use this cookie at the start of the next higher-level operation. The higher-level operation is a complete [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16) that improves the client's [**up-to-date vector**](#gt_42564a26-2ae7-41a2-a67c-3c74381d8538). See section [4.1.10.1](#Section_eac7ee183630449daefed7108d85b4cc) for an explanation of replication cycles.
  + [IDL\_DRSGetNT4ChangeLog](#Section_6e000eb660fd4d6cae82bb6479df02fa): In this instance of the "cookie" pattern, the server returns a "cookie" in the response that completes the higher-level operation. The client does not use this cookie at the start of the next higher-level operation. The client supplies the designated null value at the start of the next higher-level operation. The higher-level operation is the retrieval of a complete Windows NT 4.0 change log. See the informative summary of this method in section [4.1.11.3](#Section_7b9d7492f8094122b4fb4da26af8bbd9).
* Successfully processing an [IDL\_DSAPrepareScript](#Section_749197848e574cf5840f6f1bd226cf02) request generates a password and stores that password locally on the server. The server returns this password in the IDL\_DSAPrepareScript response. When a server is in this state (that is, when it holds the password created by IDL\_DSAPrepareScript), it processes an [IDL\_DSAExecuteScript](#Section_1cb59761aeae4f448f9e06ae75ae45ca) request that includes this password; otherwise it rejects the request.
* [IDL\_DRSInitDemotion](#Section_faca17da3f7f409298dbfd2ce7d98b8c) is called before the other demotion methods: [IDL\_DRSReplicaDemotion](#Section_8a2f0388bdfb4519a8c3384f27c11639) and [IDL\_DRSFinishDemotion](#Section_0bf530e81be04f48b8c2208031a8725f).
* Otherwise, all method requests are independent, apart from their dependencies on the state of the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9). The potential dependencies are varied, and understanding them requires understanding the state model specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1. Here are some examples:
  + Successfully processing an [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) request can create a [**crossRef object**](#gt_353fac65-0774-4ba8-8081-eb4c963f94e7). When the directory is in this state (that is, when it holds the crossRef object), an [IDL\_DRSRemoveDsDomain](#Section_aa3cfa46c737425aae65ecaf9efe7e84) request can successfully remove that crossRef object (subject to other conditions specified with IDL\_DRSRemoveDsDomain).
  + Successfully processing an IDL\_DRSAddEntry request can create an [**nTDSDSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8). When the directory is in this state (that is, when it holds the nTDSDSA object), an [IDL\_DRSRemoveDsServer](#Section_d5c310ae347a49d4a78e6ffb2eecd581) request can successfully remove that nTDSDSA object.
  + Successfully processing an [IDL\_DRSReplicaAdd](#Section_7219df914eea494f88e3780d40d2d559) request creates a repsFrom value on a server. When a server is in this state (that is, when it holds the repsFrom value), it has the information needed to make an IDL\_DRSGetNCChanges request on the DC that is specified in IDL\_DRSReplicaAdd.
  + Successfully processing an [IDL\_DRSUpdateRefs](#Section_a273bbcfaeca46088ad4127d3e597cd4) request creates a repsTo value on a server. When a server is in this state (that is, when it holds the repsTo value), it has the information needed to make an [IDL\_DRSReplicaSync](#Section_25c71d91051f4c26977fa70892f29b00) request on the DC that is specified in IDL\_DRSUpdateRefs.
  + Successfully processing an IDL\_DRSRemoveDsDomain request first requires the removal of the metadata for all DCs hosting the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) for the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) that is to be removed. This precondition is achieved by the client calling IDL\_DRSRemoveDsServer for each such DC.

State-based sequencing issues also exist between methods specified in this document and [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d), because LDAP provides another way to change the state of the directory.

* One method, [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47), has a parameter of both input and output, *dwEnumerationContext*. This parameter is defined for the following:
  + *dwInVersion*=2, and
  + InfoType=DS\_REPL\_INFO\_METADATA\_FOR\_ATTR\_VALUE, or DS\_REPL\_INFO\_METADATA\_2\_FOR\_ATTR\_VALUE, or DS\_REPL\_INFO\_CURSORS\_2\_FOR\_NC, or DS\_REPL\_INFO\_CURSORS\_3\_FOR\_NC.

For the first call to this method for a specific InfoType, the client sets *dwEnumerationContext* in *pmsgIn* to zero. The server returns an implementation-specific value for *dwEnumerationContext* in *pmsgOut*. On subsequent calls to this method with the same InfoType, the client sets the input *dwEnumerationContext* in *pmsgIn* to the last value of that field returned from the server. The purpose of this field is to allow the client to gather all the requested information, but in more than one server call. The final call is identified when the method returns ERROR\_NO\_MORE\_ITEMS. See the server implementation section for IDL\_DRSGetReplInfo ([4.1.13.3](#Section_caa2119711be4efeb9ae35477ee68c8c)) for exact details.

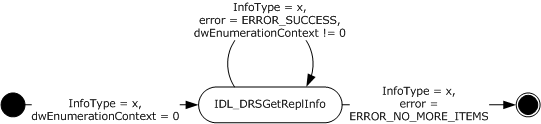


Figure 1: Using dwEnumerationContext

### Most Frequently Used Types

The role of the [DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) type, described in the previous section ([Sequencing Issues](#Section_67c5a415a6c740988cf36ef8d173cfe8)), plays a central role in capability negotiation, as explained in the specification of IDL\_DRSBind.

The type that is most central to this protocol is [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86). DSNAME is the [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the abstract DSNAME specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.1.5. A DSNAME identifies an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). Nearly every method in the DRS protocol contains a DSNAME either in its request or its response.

Another basic type in the DRS Remote Protocol is [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b). An ENTINF structure contains the DSNAME of an object (or object to be) and a list of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) with associated values–[ATTRBLOCK](#Section_f81324b8640041b5bc255117589c602a) ENTINF and ATTRBLOCK are used in the following ways:

* To communicate some or all of the current state of an object:
  + In the response to an IDL\_DRSGetNCChanges request, where multiple ENTINF structures are embedded in a [REPLENTINFLIST](#Section_c38b0412cf004b0cb4f44662a4484a00) structure. The request plus the [**stamps**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) on the object determine what information about the object is included in the response.
  + In the response to an IDL\_DRSVerifyNames request, which includes an ENTINF structure. The request specifies what information about the object is included in the response.
* To specify the state of an object to be created by a method:
  + In an IDL\_DRSAddEntry request, where the request message is essentially an ENTINF structure but is not declared as such: The DSNAME and the ATTRBLOCK structures are separate top-level fields of the request.
  + In an IDL\_DRSInterDomainMove request, where an ENTINF structure specifies the state of the object that is being created in another [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) (unlike [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) Add, with a specified objectGUID).
* To specify the subset of the current state of an object to be returned in a response:
  + In an IDL\_DRSVerifyNames request, where an ATTRBLOCK structure specifies attributes but not their values.

## Relationship to Other Protocols

This protocol includes [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) that is based on the IP protocol, which is implemented as the IDL\_DRSGetNCChanges method (section [4.1.10](#Section_B63730AC614C431C950128D6ACA91894)). [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) also supports replication based on the SMTP protocol; SMTP-based replication is specified in [[MS-SRPL]](%5bMS-SRPL%5d.pdf#Section_ec69eea50d5e428ab5bc66732aaeb866).

Some of the Active Directory state exposed by this protocol is also exposed by the Active Directory implementation of [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d); see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.

Some methods in this protocol are exposed, in modified form, via LDAP. The LDAP versions are specified in [MS-ADTS] section 3.1.1.3.

* RootDSE constructed attributes: msDS-ReplAllInboundNeighbors, msDS-ReplConnectionFailures, msDS-ReplLinkFailures, msDS-ReplPendingOps, msDS-ReplAllOutboundNeighbors, msDS-ReplQueueStatistics (these expose some functionality of IDL\_DRSGetReplInfo), dnsHostName, dsServiceName, isGlobalCatalogReady, serverName (these expose some functionality of IDL\_DRSDomainControllerInfo).
* RootDSE modify operations: becomeDomainMaster, becomeInfrastructureMaster, becomePdc, becomeRidMaster, becomeSchemaMaster, replicateSingleObject, removeLingeringObject. The last two operations expose some functionality of IDL\_DRSGetNCChanges.
* Object constructed attributes: canonicalName (this exposes some functionality of IDL\_DRSCrackNames), msDS-NCReplInboundNeighbors, msDS-NCReplCursors, msDS-ReplAttributeMetaData, msDS-ReplValueMetaData, msDS-NCReplOutboundNeighbors (these expose some functionality of IDL\_DRSGetReplInfo), tokenGroups, tokenGroupsNoGCAcceptable, tokenGroupsGlobalAndUniversal (these expose some functionality of IDL\_DRSGetMemberships and IDL\_DRSGetMemberships2).
* Controls: LDAP\_SERVER\_DIRSYNC\_OID

The LDAP control LDAP\_SERVER\_CROSSDOM\_MOVE\_TARGET\_OID is related to IDL\_DRSInterDomainMove in that the LDAP client specifies via this control the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) whose IDL\_DRSInterDomainMove method should be called (from the LDAP server implementation of Modify [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b)) to perform the move.

Some methods in this protocol have completely functional equivalents in LDAP:

* The function of IDL\_DRSWriteSPN can be performed as an LDAP Modify of the servicePrincipalName [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).
* The function of creating a crossRef [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) with IDL\_DRSAddEntry can be performed as an LDAP Add of a crossRef object.

## Prerequisites/Preconditions

This protocol is based on [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) and therefore has the prerequisites identified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) as common to all RPC interfaces.

Security configuration for usage of RPC is described further in section [2.2](#Section_0a156712918047bab0802e285f127a7f).

The [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) service must be fully initialized as described in [[MS-ADOD]](%5bMS-ADOD%5d.pdf#Section_5ff67bf4c14548cb89cd4f5482d94664) section 2.6.

## Applicability Statement

This protocol is appropriate for replicating ([**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd)-to-DC only) and managing [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in a [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) service and for overall management of the directory service.

## Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

* **Supported Transports**: [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) can be implemented on top of TCP and other protocol sequences as described in section [2.1](#Section_e6076eab53f64aad9041888c5734b715).
* **Protocol Versions:** Each of the protocol interfaces described in this document has a single version number: 4.0 for drsuapi and 1.0 for dsaop.
* **Security and Authentication Methods**: See [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 1.7.
* **Capability Negotiation**: This protocol does explicit capability negotiation as described in [IDL\_DRSBind (section 4.1.3)](#Section_605b1ea19cdc428fab7a70120e020a3d) behavior.

## Vendor-Extensible Fields

This protocol uses Win32 error codes as defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.2. Vendors SHOULD reuse those values with their indicated meaning. Choosing any other value risks a collision in the future.

This protocol uses NTSTATUS values as defined in [MS-ERREF] section 2.3. Vendors can choose their own values for this field, as long as the C bit (0x20000000) is set, indicating that it is a customer code.

## Standards Assignments

| Parameter | Value | Reference |
| --- | --- | --- |
| [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) for drsuapi methods | e3514235-4b06-11d1-ab04-00c04fc2dcd2 | Section [4.1.1](#Section_06764fc54df64104b6afa92bdaa81f6e) – section [4.1.29](#Section_ef0bfb1d037b4626a6d9cc7589bc5786) |
| RPC interface UUID for dsaop methods | 7c44d7d4-31d5-424c-bd5e-2b3e1f323d22 | Section [4.2.1](#Section_749197848e574cf5840f6f1bd226cf02) – section [4.2.2](#Section_1cb59761aeae4f448f9e06ae75ae45ca) |

# Message Transport

The following sections discuss [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a) and security considerations for this protocol. Common data types are defined and discussed in section [5](#Section_c5d9026516534ecca0d7cac691e2d08e). See section [3](#Section_c56432ffaf884443b500eecb3047da4d) for more details about the organization of this protocol specification.

## RPC Transport

This protocol uses the following [**RPC protocol sequence**](#gt_0c171cc7-e9c4-41b6-95a9-536db0042c7a): [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) over TCP as defined in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). A server MAY listen on additional RPC protocol sequences. A client SHOULD attempt to connect using the RPC-over-TCP protocol sequence.[<1>](#Appendix_A_1" \o "Product behavior note 1)

This protocol uses RPC [**dynamic endpoints**](#gt_46da887f-3f66-4941-a854-e51c52cf4c56), as described in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) part 4.

Implementations MUST use the [**UUIDs**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) as specified in section [1.9](#Section_063618edb2e24983ab133ed056700641). The RPC version number is 4.0 for the drsuapi interface and 1.0 for the dsaop interface.

## Protocol Security

This section describes the security mechanisms used for this [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331)-based protocol.

### General Background

A [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) authenticates using its [**service account**](#gt_567b7248-2521-4f32-b443-a68955906bb6). In [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024), this is the Local System account on the machine running the DC, represented by the computer [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of the machine. In [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab), a DC's service account is configured by the administrator.

In AD DS, connections for DC-to-DC communications MUST use mutual [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) and encryption of protocol traffic. Mutual authentication is provided by Kerberos (see [[MS-KILE]](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9) section 3.3.1).

In AD LDS, mutual authentication for DC-to-DC communications is not required. When a connection is established, the client uses the GSS Negotiate protocol, which first attempts to use Kerberos, and if Kerberos is unavailable, attempts NTLM (which does not give mutual authentication). If the msDS-ReplAuthenticationMode [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625) root equals 2, all DCs in the AD LDS [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) require mutual authentication for DC-to-DC communications.

Connections from a non-DC client to a DC do not require mutual authentication. Therefore, NTLM is an acceptable [**security provider**](#gt_05fd3925-0672-4f24-9dd9-2b9d441eb333) in addition to Kerberos.

When a connection is established, the non-DC client uses the GSS Negotiate protocol, which first attempts to use Kerberos and then, if Kerberos is unavailable, attempts NTLM (which does not give mutual authentication).

### Service Principal Names for Domain Controllers

In the absence of a trusted naming service, which maps service names to servers providing a given service, the client of a distributed service must authenticate a *service*, not a server. The client produces a [**service principal name (SPN)**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4), which is a name for the service it wants a connection to, and the [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) system verifies that the server is a provider of the named service.

Kerberos verifies the services provided by a server by reading the servicePrincipalName [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of the server's computer [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). The servicePrincipalName attribute contains a set of [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) strings; each string is an SPN. If the client produces an SPN that is not present on the computer object of the server it has requested a connection to, the mutual authentication fails and so does the connection attempt.

Each [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) maintains the values of the servicePrincipalName attribute on its own computer object.

For the protocols specified in this document, the SPN produced by a client differs for DC-to-DC communications and non-DC-client-to-DC communications. The specific SPNs produced by a client in each scenario are described in the following sections.

In either DC-to-DC or client-to-DC operations, to allow use of the DRS Remote Protocol when the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) endpoint mapper has been configured to disallow anonymous clients (see [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.1.1.1.3), the DC stores an SPN with the following format[<2>](#Appendix_A_2" \o "Product behavior note 2):

* "RPC/<DSA GUID>.msdcs.<DNS forest name>"

In the preceding SPN description, <DSA GUID> is the [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the DC and <DNS forest name> is the [**FQDNs (2)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) of the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) of the DC.

### DC-to-DC Operations

This section describes the security and mutual [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) requirements for those DRS Remote Protocol operations that involve [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd)-to-DC interactions.[<3>](#Appendix_A_3" \o "Product behavior note 3)

#### Security Provider

If mutual [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) is required, a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) client MUST request authentication, specifying the "Kerberos" [**security provider**](#gt_05fd3925-0672-4f24-9dd9-2b9d441eb333) (RPC\_C\_AUTHN\_GSS\_KERBEROS). Regardless of whether mutual authentication is required, a DC client MUST request integrity and encryption of the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) messages by specifying an [**authentication level**](#gt_bfb9708e-9d05-4f79-8969-ef63f73aa434) (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 2.2.1.1.8) of "packet privacy" (RPC\_C\_AUTHN\_LEVEL\_PKT\_PRIVACY).

A DC client MUST authenticate the target DC by constructing an [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) for the service it is using. A DC client constructs an SPN as described in the following section.

#### SPN for a Target DC in AD DS

Two different scenarios are possible when an [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024) [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) wants to connect to another DC for a DRS protocol operation:

* A DC wants to connect to a DC in a particular [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).
* A DC wants to connect to a [**GC server**](#gt_a5a99ce4-e206-42dc-8874-e103934c5b0d) (see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.1.10) in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62).

The scenario determines how the DC constructs an [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) for the service it is using:

* A DC wants to connect to a DC in a particular domain. The DC constructs the following SPN:
  + "<DRS interface GUID>/<DSA GUID>/<DNS domain name>"
* A DC wants to connect to a GC server in the forest. The DC constructs the following SPN:
  + "GC/<DNS hostname>/<DNS forest name>"

In the preceding SPN descriptions:

* "GC" is a literal string that represents a [**service class**](#gt_c1386afd-9d42-47c7-a7ea-d01912a17647).
* The forward slash ('/') is the literal separator between parts of the SPN.
* <DRS interface GUID> is the fixed DRS [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1), which has the well-known value of "E3514235-4B06-11D1-AB04-00C04FC2DCD2".
* <DSA GUID> is the [**DSA**](#gt_919e41e4-f321-43ed-bbe5-675ca4cd9a28) GUID of the target DC.
* <DNS domain name> is the [**FQDN (2)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) of the domain of the target DC.
* <DNS hostname> is the DNS host name of the target DC.
* <DNS forest name> is the FQDN (2) of the forest of the target DC.

For example, the two SPNs that can be used for a DC named "dc1" with [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) A5FF6869-AB5A-11D2-91E2-08002BA3ED3B in the contoso.com domain and forest are as follows:

* "E3514235-4B06-11D1-AB04-00C04FC2DCD2/A5FF6869-AB5A-11D2-91E2-08002BA3ED3B/contoso.com"
* "GC/dc1.contoso.com/contoso.com"

To allow mutual [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) to occur in DC-to-DC protocol operations, an AD DS [**RODC**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870) MUST store the form of SPN that begins with "GC/" as values of the servicePrincipalName [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of the DC's [**computer object**](#gt_d8e8f5a7-db85-40a8-98ed-1abab2237b82), but not the other form of SPN because that form of SPN is used for outbound [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb). Other AD DS DCs MUST store both forms of SPN as values of the servicePrincipalName attribute of the DC's computer object. Additional forms that must be stored for client-to-DC protocol operations are described in section [2.2.4.2](#Section_894d09997d794e81a4077bcf6522b0a7).

#### SPN for a Target DC in AD LDS

When an [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) wants to connect to another DC for a DRS protocol operation, it uses either of the following [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) forms:

* <DRS interface GUID>-ADAM/<DNS hostname>:<LDAP port>
* <DRS interface GUID>-ADAM/<NetBIOS hostname>:<LDAP port>

In the preceding SPN descriptions:

* <DRS interface GUID> is the fixed DRS [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1), which has the well-known value of "E3514235-4B06-11D1-AB04-00C04FC2DCD2".
* "-ADAM/" is a literal string.
* <DNS hostname> is the full DNS host name of the target DC.
* <NetBIOS hostname> is the NetBIOS host name of the target DC.
* The colon (':') is the literal separator between the host name and port number.
* <LDAP port> is the [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) port on which the target DC listens.

If an AD LDS DC runs on a machine joined to an [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), and NTDSDSA\_OPT\_DISABLE\_SPN\_REGISTRATION is not present in the options [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of its nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) ([[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.1.2.2.1.2.1.1), the AD LDS DC MUST store these two forms of SPN as values of the servicePrincipalName attribute of the object (in the external [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024) domain) that corresponds to the [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) that the AD LDS service is running as. This action allows mutual [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) to occur in DC-to-DC protocol operations. Additional forms that must be stored for client-to-DC protocol operations are described in section [2.2.4.3](#Section_3a6c821d5465414995247bec717fa60a).

### Client-to-DC Operations

This section describes the security and mutual [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) requirements for those DRS Remote Protocol operations that involve client-to-[**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) interactions.

#### Security Provider

To request [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317), a client program specifies the "GSS Negotiate" [**security provider**](#gt_05fd3925-0672-4f24-9dd9-2b9d441eb333) (RPC\_C\_AUTHN\_GSS\_NEGOTIATE). Regardless of whether mutual authentication is required, a client MUST request integrity and encryption of the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) messages by specifying an [**authentication level**](#gt_bfb9708e-9d05-4f79-8969-ef63f73aa434) of "packet privacy" (RPC\_C\_AUTHN\_LEVEL\_PKT\_PRIVACY).

To authenticate the target [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), a client program constructs an [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) for the service it is using and negotiates Kerberos as the security provider. A client constructs an SPN as described in the following sections.

#### SPN for a Target DC in AD DS

Three scenarios are possible when a client wants to connect to an [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024) [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) for a DRS Remote Protocol operation:

* A client wants to connect to a particular DC by using its host name, regardless of the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) it contains.
* A client wants to connect to a DC in a particular domain.
* A client wants to connect to a [**GC server**](#gt_a5a99ce4-e206-42dc-8874-e103934c5b0d) (see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.1.10) in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62).

The scenario determines how the client constructs an [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) for the service it is using:

* A client wants to connect to a particular DC by using its host name, regardless of the domain it contains. The client constructs any of the following three SPNs:
  + "ldap/<NetBIOS hostname>"
  + "ldap/<DNS hostname>"
  + "ldap/<DSA GUID based DNS hostname>"

The SPN that a client constructs depends on the information that the client has available. For example, some clients have only a NetBIOS name for a DC, while others have only an [**Internet host name**](#gt_4d5d5403-372f-4f9f-8d7a-65c310c807d9) for a DC.

* A client wants to connect to a DC in a particular domain. The client constructs any of the following three SPNs:
  + "ldap/<DNS hostname>/<NetBIOS domain name>"
  + "ldap/<DNS hostname>/<DNS domain name>"
  + "ldap/<NetBIOS hostname>/<NetBIOS domain name>"[<4>](#Appendix_A_4" \o "Product behavior note 4)

The SPN that a client constructs depends on the information that the client has available. For example, some clients have only a NetBIOS name for a domain, while others have only a [**fully qualified domain name (FQDN) (2)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) for a domain.

* A client wants to connect to a GC server in the forest:
  + "GC/<DNS hostname >/<DNS forest name>"

In the preceding SPN descriptions:

* "ldap" and "GC" are literal strings representing [**service classes**](#gt_c1386afd-9d42-47c7-a7ea-d01912a17647).
* The forward slash ('/') is the literal separator between parts of the SPN.
* <NetBIOS hostname> is the NetBIOS host name of the target DC.
* <DNS hostname> is the DNS host name of the target DC.
* <NetBIOS domain name> is the [**NetBIOS domain name**](#gt_f7f8efcc-c6d5-40f0-9605-c9d99c5a0b92) of the target DC.
* <DNS domain name> is the FQDN (2) of the domain of the target DC.
* <DSA GUID based DNS hostname> is the DNS host name of the target DC, constructed in the form "<DSA GUID>.\_msdcs.<DNS forest name>".
* <DNS forest name> is the FQDN (2) of the forest of the target DC or the target GC server.

As an example, the two- and three-part SPNs that can be used for a DC named "dc1" in the contoso.com domain are as follows:

* "ldap/DC1"
* "ldap/dc1.contoso.com"
* "ldap/6B352A21-8622-4F6D-A5A9-45CE9D7A5FB7.\_msdcs.contoso.com"
* "ldap/dc1.contoso.com/CONTOSO"
* "ldap/dc1.contoso.com/contoso.com"
* "GC/dc1.contoso.com/contoso.com"
* "ldap/DC1/CONTOSO"

To allow mutual [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) to occur in client-to-DC protocol operations, an AD DS DC MUST store these seven forms of SPN as values of the servicePrincipalName [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of the DC's computer [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). The [**GC**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169) SPN for client-to-DC is identical to the GC SPN for DC-to-DC. Therefore, when the requirements of this section are added to the requirements of section [2.2.3.2](#Section_41efc56e00074e88bafed7af61efd91f), an AD DS [**RODC**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870) MUST store six, and other AD DS DCs MUST store seven, servicePrincipalName values in all.

#### SPN for a Target DC in AD LDS

When a client wants to connect to an [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) for a DRS operation, it uses either of the following [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) forms:

* ldap/<DNS hostname>:<LDAP port>
* ldap/<NetBIOS hostname>:<LDAP port>

In the preceding SPN descriptions:

* "ldap" is the literal string representing the [**service class**](#gt_c1386afd-9d42-47c7-a7ea-d01912a17647).
* The forward slash ('/') is the literal separator between parts of the SPN.
* <DNS hostname> is the full DNS host name of the target DC.
* <NetBIOS hostname> is the NetBIOS host name of the target DC.
* The colon (':') is the literal separator between the host name and port number.
* <LDAP port> is the [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) port on which the target DC listens.

If an AD LDS DC runs on a computer joined to an external [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), and NTDSDSA\_OPT\_DISABLE\_SPN\_REGISTRATION is not present in the options [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of its [**nTDSDSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8) in AD LDS (see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.1.2.2.1.2.1.1), then the AD LDS DC MUST store these two forms of SPN as values of the servicePrincipalName attribute of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) (in the external [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024) domain) that corresponds to the [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) that the AD LDS service is running as. This action allows mutual [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) to occur in "client-to-AD LDS DC" protocol operations. When the requirements of this section are added to the requirements of section [2.2.3.3](#Section_debb73a41e5149f3ac62ae49ce35d13f), an AD LDS DC that stores SPNs stores four servicePrincipalName values in all.

## Directory Service Schema Elements

This protocol is part of the [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) core family of protocols. To be fully compliant with Active Directory, an implementation of this protocol must be used in conjunction with the full Active Directory [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093), containing all the schema [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) and [**classes**](#gt_18393bbe-0c06-42b7-890d-b94a9a40b6e0) specified in [[MS-ADA1]](%5bMS-ADA1%5d.pdf#Section_19528560f41e4623a406dabcfff0660f), [[MS-ADA2]](%5bMS-ADA2%5d.pdf#Section_e20ebc4e528540bab3bdffcb81c2783e), [[MS-ADA3]](%5bMS-ADA3%5d.pdf#Section_4517e8353ee644d4bb95a94b6966bfb0), [[MS-ADLS]](%5bMS-ADLS%5d.pdf#Section_9427994325ab4c139bf26d411cc2f796), and [[MS-ADSC]](%5bMS-ADSC%5d.pdf#Section_9abb5e97123d4da99557b353ab79b830).

# Background to Behavior Specifications

## Document Organization

In this document, information that is relevant to each particular [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) method is grouped with the specification of the behavior for that method. Information that is relevant to a particular RPC method includes: the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definition of that method, definitions for all types used exclusively by that method, and examples specific to that method.

Most methods specified in this document have no special client considerations. In such cases, the entire specification of the method behavior is the specification of server behavior.

In cases where client behavior is specified, the client behavior in preparing a request is specified in the section immediately preceding the section that specifies server behavior, and the client behavior in processing a response is specified in the section immediately following the section that specifies server behavior. This ordering follows the flow of processing a request.

The behavior specification for some methods is followed immediately by one or more examples that show a request as seen by the server implementation of the method, the corresponding response created by the server implementation of the method, and the effect of server request processing on the state of the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9). Section [3.5.1](#Section_f200c87ecd674da4a45258a4761c8970) specifies the initial state used by all examples.

In cases where a type used in a method request or response is common to several methods, that type is placed in [Common Data Types, Variables and Procedures (section 5)](#Section_c5d9026516534ecca0d7cac691e2d08e). This section is arranged alphabetically, and so its table of contents serves as an index. This section is placed after the section that contains method behavior specifications, because typically a reader will reference the common types while reading the method specifications, and not the other way around.

## Typographical Conventions

Sections of this document are not self-contained; they contain both forward and backward references, all of which are hyperlinked. In addition, the following typographical convention is used to indicate the special meaning of certain names:

* Underline, as in instanceType: the name of an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) or [**object class**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a) whose interpretation is specified in the following documents:
  + [[MS-ADA1]](%5bMS-ADA1%5d.pdf#Section_19528560f41e4623a406dabcfff0660f) Attribute names whose initial letter is A through L.
  + [[MS-ADA2]](%5bMS-ADA2%5d.pdf#Section_e20ebc4e528540bab3bdffcb81c2783e) Attribute names whose initial letter is M.
  + [[MS-ADA3]](%5bMS-ADA3%5d.pdf#Section_4517e8353ee644d4bb95a94b6966bfb0) Attribute names whose initial letter is N through Z.
  + [[MS-ADSC]](%5bMS-ADSC%5d.pdf#Section_9abb5e97123d4da99557b353ab79b830) Object class names.
  + [[MS-ADLS]](%5bMS-ADLS%5d.pdf#Section_9427994325ab4c139bf26d411cc2f796) Object class names and attribute names for [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab).

No special typographical convention is used for names that represent elements of sets; for example, DRS\_WRIT\_REP. The name of the set type (for example, [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030)) is always clear from context, and the elements of each set type are defined with the set type. Similarly, no special typographical convention is used for names that represent [**Windows error codes**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b); for example, ERROR\_INVALID\_PARAMETER.

## State Model

### Preliminaries

[[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.1 is a prerequisite to the remainder of this specification.

### Transactions

The specifications of client and server method behavior in this document do not mention transaction boundaries because all methods use transactions in a systematic way, as described in the remainder of this section.

In server processing of a normal method, a transaction begins implicitly on the first access to the database that represents the persistent state of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), and ends implicitly before a method returns. When a new logical thread of control is created (see Asynchronous Processing in section [3.4.6](#Section_9d615626ace2445dadfbc9189c1599be)), the originating thread implicitly ends its transaction before it returns, and the new logical thread of control implicitly begins an unconnected transaction as described above.

If a transaction fails, and the method return would otherwise have been successful, the [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) returned by the method is in one of the following sets:

* **Retryable**: ERROR\_DS\_DRA\_BUSY, ERROR\_DS\_OUT\_OF\_VERSION\_STORE. There is a significant chance that retrying the request will succeed.
* **Implementation limit**: ERROR\_DS\_MAX\_OBJ\_SIZE\_EXCEEDED. This error is returned when an implementation-specific, fixed size limit is exceeded. Retrying will not succeed, but the system is functioning normally.
* **Resource limit**: ERROR\_DISK\_FULL, ERROR\_NO\_SYSTEM\_RESOURCES. Retrying will not succeed; an administrator must increase available resources.
* **Corruption**: ERROR\_DS\_KEY\_NOT\_UNIQUE, ERROR\_DS\_OBJ\_NOT\_FOUND, ERROR\_DISK\_OPERATION\_FAILED. Retrying will not succeed; an administrator must repair the database that represents the persistent state of the DC or restore the database from backup.

If server processing of a normal method performs some [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) and then detects an error condition, it terminates the current transaction before returning the error code that describes the error condition. If the transaction termination encounters an error condition, the method does not report the transaction-related error condition. Instead, the method reports the original error condition.

When the specification includes a client preparing a method request or processing a method response, the pattern is similar. When a client that is a DC prepares a method request, it implicitly begins a transaction on the first access to the database that represents the persistent state of the client DC and commits this transaction before sending the request. When a client that is a DC processes a method response, it implicitly begins a transaction on the first access to the database that represents the persistent state of the DC and commits this transaction when processing of the response is complete. A client transaction is never in progress while the client waits for the server to respond to a method request. There is no use of distributed transactions.

### Concrete and Abstract Types

This protocol specification involves both [**concrete**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) and [**abstract types**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b).

A concrete type is a type whose representation must be standardized for interoperability. In this protocol specification, three cases apply:

* Types in the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definition of the drsuapi and dsaop [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interfaces that determine the format of network requests and responses.
* Types that are hand marshaled onto the network, such as types that are sent in drsuapi and dsaop requests and responses as octet strings whose actual structure is hidden from the IDL compiler. The hand marshaling and corresponding hand unmarshaling are performed by the implementation of [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) and by clients of the drsuapi and dsaop RPC interfaces.
* Types that are hand marshaled into [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f), such as types that are stored in the directory as octet strings. The hand marshaling and corresponding hand unmarshaling are performed by the implementation of Active Directory and by clients of the Active Directory [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) interface [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.3.

Concrete types in the first category are specified by the C / IDL type declaration. Concrete types in the second and third categories are specified pictorially. Some types are in multiple categories and are specified both ways.

All other types in the specification are *abstract*, meaning that their use is internal to the specification. Abstract types are based on the standard mathematical concepts set, sequence, directed graph, and tuple.

This specification introduces the notion of an abstract attribute. An *abstract attribute* is an Active Directory attribute that has an abstract type for use in pseudocode. An abstract attribute can have a specified concrete representation, required for interoperability; in that case, the abstract attribute's type definition specifies the correspondence between information in the abstract type and in the concrete type. This relieves the specification pseudocode from concerns with storage allocation, packing variable-length information into structures, and so on.

Pseudocode deals with a mixture of concrete and abstract types. The notations and conventions for each are specified in section [3.4](#Section_887dc359a000459f806f82f8a06eb9cc).

## Pseudocode Language

### Naming Conventions

Identifiers for [**concrete types**](#gt_cd539538-9f7e-4881-b5af-2301b420244d), structure fields, and constants are used unchanged. The names of concrete types are often uppercase, with underscore characters ('\_') to mark the divisions between words.

* Examples: REPS\_FROM, DRS\_MSG\_UPDREFS

Identifiers for [**object classes**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a) and [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) are [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) display names from [[MS-ADA1]](%5bMS-ADA1%5d.pdf#Section_19528560f41e4623a406dabcfff0660f), [[MS-ADA2]](%5bMS-ADA2%5d.pdf#Section_e20ebc4e528540bab3bdffcb81c2783e), [[MS-ADA3]](%5bMS-ADA3%5d.pdf#Section_4517e8353ee644d4bb95a94b6966bfb0), and [[MS-ADSC]](%5bMS-ADSC%5d.pdf#Section_9abb5e97123d4da99557b353ab79b830). These identifiers start with a lowercase letter; there are no capitalization conventions for the letters that follow the initial lowercase letter. These identifiers, used in this document to improve the readability of the examples, are equivalent to the [ATTRTYP (section 5.14)](#Section_9117312908e6497c8266b5ac0aa5f983) that actually identifies object classes and attributes. The mapping between ATTRTYP and the [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093) [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) representing a [**class**](#gt_18393bbe-0c06-42b7-890d-b94a9a40b6e0) or attribute is specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.2.6.

* Examples: repsFrom, nTDSDSA

Identifiers for types and procedures introduced for specification purposes always start with an uppercase letter, and start each word after the first word with an uppercase letter (Pascal case).

* Examples: RepsFrom, ValidateSiteRDN

Identifiers for variables introduced for specification purposes always start with a lowercase letter, and start each word after the first word with an uppercase letter (camel case).

* Examples: dc, vSet

### Language Constructs for Concrete Types

[**Concrete types**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) support structure assignments between types that are not identical. For example:

1. reqV1: DRS\_MSG\_REPADD\_V1
2. reqV2: DRS\_MSG\_REPADD\_V2
3. reqV2 := reqV1

Such an assignment is shorthand for a field-by-field assignment for fields with the same name in the two structures. The preceding example is equivalent to the following:

1. reqV1: DRS\_MSG\_REPADD\_V1
2. reqV2: DRS\_MSG\_REPADD\_V2
3. reqV2.pNC := reqV1.pNC
4. reqV2.rtSchedule := reqV1.rtSchedule
5. reqV2.ulOptions := reqV1.ulOptions

The ADR built-in function returns the address of a variable. The ADDRESS OF type constructor creates a pointer type. These are needed occasionally when dealing with concrete structures.

Pseudocode does not perform storage allocation for concrete response structures. An implementation is free to allocate any amount of memory sufficient to contain the structures within the response.

### Language Constructs for Abstract Types

The language includes the conventional types *Boolean* and *Integer*.

The notation [first .. last] stands for the *subrange* first, first+1, ... , last. The type *byte* is the subrange [0.. 255].

A *sequence* is an indexed collection of variables, called the *elements* of the sequence. The elements all have the same type. The *index type* of a sequence is a zero-based subrange. *S*[*i*] denotes the element of the sequence *S* that corresponds to the value *i* of the index type. The number of elements in a sequence *S* is denoted *S*.length. Therefore, the index type of a sequence *S* is [0 .. *S*.length-1].

A sequence type can be *open* (index type not specified) or *closed* (index type specified):

* type DSNameSeq = sequence of [DSName](#Section_a0d5477a522946b9890a54b924d487d1)
* type Digest = sequence [0 .. 15] of byte

A fixed-length sequence can be constructed by using the following notation:

* [*first element*, *second element*, ... , *last element*]

Therefore:

* *s* := []

sets a sequence-valued variable s to the empty sequence. A sequence of bytes can be written in the more compact string form shown in the following example:

* *s* := "\x55\x06\x02"

A *unicodestring* is a sequence of 16-bit [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) characters.

If *S* is a sequence, and *j* ≥ *i*, then *S*[*i* .. *j*] is a new sequence of length *j* - *i* + 1, whose first element has value *S*[*i*], second element has value *S*[*i* + 1], ... , and final element has value *S*[*j*]. The index set of the new sequence is [0 .. *j* - *i*]. If *j* < *i* then *S*[*i* .. *j*] is the empty sequence.

A *tuple* is a set of name-value pairs: [name1: value1, name2: value2, ... , namen: valuen] where namek is an identifier and valuek is the value bound to that identifier. Tuple types are defined as in the following example:

* type DSName = [dn: [DN](#Section_837c7001335148dea177c165a584816c), guid: [GUID](#Section_5e740f50e6a048c9bca800072e85d963), sid: [SID](#Section_13560cc227ff43a09d6fd686bccc5f3c)]

This example defines DSName as a tuple type with a DN-valued field dn, a GUID-valued field guid, and a SID-valued field sid.

A *tuple constructor* is written as in this example:

* dsName: DSName
* dsName := [dn: "cn=Peter Houston,ou=NTDEV,dc=microsoft,dc=com"]

Fields that are unspecified in a tuple constructor are assigned null values in the resulting tuple.

Access to the named fields of a tuple uses dot notation. Continuing the example:

* d: DN; g: GUID; s: SID
* d := dsName.dn
* g := dsName.guid
* s := dsName.sid

The preceding assignments set the variable *d* to "cn=Peter Houston,ou=NTDEV,dc=microsoft,dc=com", and variables *g* and *s* to null values.

A *tuple deconstructor* can be written anywhere a tuple-valued variable can occur. The preceding assignments are equivalent to the following:

* [dn: d, guid: g, sid: s] := dsName;

The language includes *sets*. If *S* is a set, number(*S*) is the cardinality of the set *S*.

A fixed-size set can be constructed using the notation:

* {*one element*, *another element*, ... , *yet another element*}

Therefore:

* *S* := {}

sets a set-valued variable *S* to the empty set.

If *S* is a set, the predicate *x in S* is true if *x* is a member of *S*. Therefore, the value of the expression:

* 13 in {1, 2, 3, 5, 7, 11}

is false.

If A and B are sets, A + B is the set union of A and B, A ∩ B is the set intersection of A and B, and A - B is the set difference of A and B.

The specification uses [KNUTH1] section 2.3.4.2 as a reference for the graph-related terms *directed graph*, *oriented tree*, *vertex*, *arc*, *initial vertex*, and *final vertex*. In pseudocode, graphs are described in terms of their vertex and arc sets, and individual arcs are represented as tuples.

The language supports coercion between [**abstract**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) and [**concrete types**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) when the correspondence between the two is clear. For example, if *stringSet* is a set of unicodestring and *stringArrayPtr* is a pointer to an array of pointers to null-terminated Unicode strings, the assignment:

* stringSet := stringArrayPtr^

populates the abstract set of strings by copying from the concrete array of strings.

### Common Language Constructs

The syntax of standard control structures:

1. if boolean-expr then
2. stmts
3. else
4. stmts
5. endif
6. if boolean-expr then
7. stmts
8. else if boolean-expr then : disambiguated by indentation
9. stmts
10. endif
11. foreach var in set-or-sequence-expr
12. stmts
13. endfor
14. for var := first-value to last-value
15. stmts
16. endfor
17. while boolean-expr
18. stmts
19. endwhile
20. return expr
21. raise expr

The keyword **raise** is used to raise an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) exception. The operand of the **raise** expression specifies the RPC exception to be raised. Details of how an RPC exception is raised are specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 12.6.4.7.

Other constructs used (inspired by Modula-3; for more information, see [NELSON]):

1. : declare a procedure
2. : with typed args and result
3. procedure name(arg: type, arg: type, ... , arg: type): type
4. : declare a procedure
5. : with call-by-reference args
6. procedure name(var arg: type, ... , var arg: type): type
7. : cast a variable or an expression value
8. : to a different type
9. loophole(expr, type)
10. var: type : declare a variable with a type
11. var := expr : assignment
12. expr^ : pointer dereferencing
13. expr.id : field selection

List of infix and prefix operator binding precedence (strongest binding at the top of the list):

1. x.a : infix dot
2. f(x) a[i] : applicative (, [
3. p^ : postfix ^
4. + - : prefix arithmetics
5. \* / mod ∩ : infix arithmetics; set intersection
6. + - : infix arithmetics; set union and difference
7. = ≠ < ≤ ≥ > in : infix relations
8. not : prefix not
9. and : infix and
10. or : infix or

All infix operators are left-associative, and so, for example:

1. a - b + c

means:

1. (a - b) + c

Parentheses can be used to override the precedence rules.

The infix Boolean operators "and" and "or" are evaluated left to right, conditionally. The expression "p and q" is true if both p and q are true. If p is false, q is not evaluated. The expression "p or q" is true if at least one of p and q are true. If p is true, q is not evaluated.

### Access to Objects and Their Attributes

The specification contains many accesses to specific [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f). The specification uses the following concise notation for these accesses to aid readability. If o is a variable that contains a [DSName](#Section_a0d5477a522946b9890a54b924d487d1) or a [DN](#Section_837c7001335148dea177c165a584816c), then:

1. o!attr

... is an access to the attr attribute of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) named by the content of o, performed in the context of the [**NC replicas**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) held by the server. In this notation, the name attr is a constant (like objectGUID), not a variable.

If the form o!attr occurs in an expression context, it denotes a value. There are three possibilities:

* If the attr attribute is not present on o, the value of the expression is the distinguished value null.
* If the attr attribute is present and declared multivalued, the value of the expression is a *set* that contains all the values of attr. If only one value is present, the value of the expression is a set that contains one element, the value.
* If the attr attribute is present and declared single-valued, the value of the expression is the value of attr.

If the form o!attr occurs on the left side of an assignment statement, it is used as a variable. The attr attribute need not already be present on o for this assignment to be well defined. The assignment:

1. o!attr := null

... removes the attr attribute from object o.

The distinguished value null is an admissible value for any type that is stored as the value of an attribute. Suppose, for example, that attr is a single-valued integer attribute. If attr is not present on object o, the assignment:

1. i := o!attr

... assigns the value null to the integer variable i. There is no ambiguity between this use of null and the use of null as the value of a pointer, because pointer values cannot be stored as the value of an attribute.

The value null can be used in the following ways:

* Tested for equality or inequality.
* Used where a sequence value is expected; it is equivalent to [], the empty sequence.
* Used where a set value is expected; it is equivalent to {}, the empty set.
* Used within a set constructor, where it adds no element to the resulting set.

The value null cannot be used in other expressions involving normal values. Therefore:

1. i: integer
2. s: set
3. i := o!attr
4. s := { o!attr }
5. if i = null then /\* attr not present on object o \*/
6. s := s + o!attr
7. endif

... is a valid pseudocode sequence. If the attr attribute is not present on object o, the branch of the if statement will be executed, and the set s is empty. But the statement:

1. i := o!attr \* 2

... is a specification error if the attr attribute is not present on object o.

Queries in this specification are expressed in one of the following two forms:

1. rt := select all scope where predicate
2. rt := select one scope where predicate

In either form of query, *scope* specifies the set of values or objects to be examined, and *predicate* specifies the subset of the scope that is the query result.

Scopes take the form:

1. var from set-of-values-or-objects

... where *var* is an identifier to be used in the predicate, and *set-of-values-or-objects* is a set of values or DSNames that designate objects. These sets can be the result of evaluating any expression; for example, they can be the values of local set-valued variables. But usually they are sets of values or objects from the directory; for example, in the following form:

1. var from o!attr

... the scope is the set of all values of attribute attr on object o; by the definition of null, the scope is the empty set if o!attr = null.

There are three special forms for scopes that are sets of objects:

1. var from children o
2. var from subtree o
3. var from all

Here, *o* is a DSName or DN valued variable. The form children o denotes the set of children of the object *o* within the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) of *o*. This form does not include the object *o* itself. The form subtree o denotes the set of all descendants of *o* within the NC of *o*, plus the object *o* itself. The form all denotes the set of all objects in all NC replicas held by the server.

The predicate is an arbitrary predicate that uses the scoping identifier (*var*, noted earlier) as a variable. The query is evaluated by binding each value or object (in arbitrary order) to *var*, and then evaluating the predicate. If the predicate is true, the value or object is said to *satisfy* the predicate.

If the query takes the form "select all", the result of the evaluation is the set of all values or objects in the scope that satisfy the predicate. If the scope is a set of values, the type of the result is a set of values; otherwise, the type of the result is a set of DSName.

If the query takes the form "select one", the result of the evaluation is any single value or object that satisfies the predicate, or null if no value or object satisfies the predicate. If more than one result is possible, the result is nondeterministic. If the scope is a set of values, the type of the result is the type of the value; otherwise, the type of the result is DSName.

Here is a query example:

1. rt := select one v from nc!repsTo where
2. v.naDsa = pReq^.V1.pszDsaDest or
3. v.uuidDsa = pReq^.V1.uuidDsaObjDest
4. if rt = null then
5. /\* no matching values \*/
6. endif

In the "children / subtree / all" forms, as specified, the scope includes normal objects, not [**tombstones**](#gt_9d8e0963-13fa-4e19-a97f-7ce6bc90d20f). Adding the qualifier "-ts-included" to these forms expands the scope to include both normal objects and tombstones. For example, the expression:

1. select all o from subtree-ts-included nc

... returns the set that contains the DSNames of all objects and tombstones in the subtree that is rooted at the DSName **nc**.

### Asynchronous Processing

Several methods involve "asynchronous processing" in which a method initiates a separate logical thread of control with some initial state, and then the method execution continues independently. However, all the documented operations are synchronous operations as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). No documented operations make use of [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331)-defined asynchronous processing.

The phrase "logical thread of control" suggests that asynchronous processing can be implemented in a variety of ways, including message processing (where each message represents a logical thread of control), "heavyweight" processes that have exclusive use of an address space, system-level multi-threading within a single address space, thread pooling, and so on.

A method that uses asynchronous processing always returns its response immediately after initiating the separate logical thread of control; there is never any interaction with the new logical thread of control. The results of the new logical thread of control are visible only through its effects on the database representing the persistent state of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). If the server crashes before the new logical thread of control has completed all its documented effects, the new logical thread of control never has any effects.

Asynchronous processing is always performed in the [**security context**](#gt_88d49f20-6c95-4b64-a52c-c3eca2fe5709) of the server itself, not the security context of the client. Therefore, all necessary access checks MUST be performed before the new logical thread of control is initiated.

This design pattern is indicated by the following text in the pseudocode:

1. Asynchronous Processing: Initiate a logical thread of control
2. to process the remainder of this request asynchronously

## Conventions for Protocol Examples

### Common Configuration

This section specifies the test setup that is used for most of the examples presented in section [4](#Section_9554afa5e7554742a34b899fc4e2fd20). The behavior of certain methods can be highlighted only by starting from a different state. The example section for such a method specifies the difference between the initial state used for that example and the state given here.

The configuration is a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) with two [**domains**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) CONTOSO.COM (Forest Root Domain) and ASIA.CONTOSO.COM (Domain NC):

Forest: CONTOSO.COM

* The forest functional level is DS\_BEHAVIOR\_WIN2003 functional level; therefore only Windows Server 2003 operating system or higher versions of [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) are present in the forest. All DCs are running Windows Server 2003 Enterprise Edition.

Domains:

* CONTOSO.COM (Forest Root Domain NC), whose NetBIOS name is CONTOSO.
* ASIA.CONTOSO.COM (Domain NC), whose NetBIOS name is ASIA.

Sites:

* Default-First-Site-Name
* Default-Second-Site-Name

DCs:

* Domain: CONTOSO.COM
  + CN=DC1, OU=DOMAIN CONTROLLERS, DC=CONTOSO, DC=COM,
  + CN=DC2, OU=DOMAIN CONTROLLERS, DC=CONTOSO, DC=COM,
* Domain: ASIA.CONTOSO.COM
  + CN=DCA1, OU=DOMAIN CONTROLLERS, DC=ASIA, DC=CONTOSO, DC=COM.

Domain-joined computer:

* Domain: CONTOSO.COM
  + CN=M1, CN=COMPUTERS, DC=CONTOSO, DC=COM.

Users added:

* Domain: CONTOSO.COM
  + CN =Kim Akers, CN =Users, DC =CONTOSO, DC =COM,
* Domain: ASIA.CONTOSO.COM
  + CN =Yan Li, CN =Users, DC = ASIA, DC =CONTOSO, DC =COM,

Groups added:

* Domain: CONTOSO.COM
  + CN =GroupA, CN =Users, DC =CONTOSO, DC =COM,
    - objectSid: S-1-5-21-254470460-2440132622-709970653-1114
    - member: null
    - groupType: {GROUP\_TYPE\_RESOURCE\_GROUP, GROUP\_TYPE\_SECURITY\_ENABLED}
  + CN = Administrators, CN =Builtin, DC =CONTOSO, DC =COM
    - objectSid: S-1-5-32-544
    - member: Domain Admins, Enterprise Admins, Local Administrator of DC1
    - groupType: {GROUP\_TYPE\_BUILTIN\_LOCAL\_GROUP, GROUP\_TYPE\_RESOURCE\_GROUP, GROUP\_TYPE\_SECURITY\_ENABLED}

### Data Display Conventions

The typical (server behavior only) example shows an initial state, a request, a response, and a final state.

The initial and final states highlight the changes for methods that perform [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493). If the method is a query, then only the initial state is shown.

States are rendered using the LDP tool. The LDP transcript shown has been edited slightly for clarity. Specifically:

* The "ld" and "&msg" are not shown for each search request. Nor is the "0" that means "typesOnly = false".
* The actual [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) list is shown, in italics, within square brackets. The LDP tool does not show it in the transcript it produces.
* The numeric constant that controls the search scope is replaced by its [[RFC2251]](https://go.microsoft.com/fwlink/?LinkId=90325) name: *baseObject*, *singleLevel*, or *wholeSubtree*.

For example, the string:

ldap\_search\_s(ld, "DC=CONTOSO,DC=COM", 0, "(objectclass=\*)", attrList, 0, &msg)

in the LDP transcript is changed to:

ldap\_search\_s("DC=CONTOSO,DC=COM", *baseObject*, "(objectclass=\*)", [*repsTo*])

assuming that the search requested that only the repsTo attribute be returned.

Requests and responses are rendered by using the Windows debugger in the context of the server (for server behavior) or client (for client behavior), with editing of the transcript for clarity. The following two edits are performed consistently:

* The [DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) parameter is not shown.
* Where the value of a parameter is a [**binary large object (BLOB)**](#gt_ad861812-8cb0-497a-80bb-13c95aa4e425), the value is not shown, but instead expressed as *binary blob*.

## Server and Client Initialization

The server MUST start the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) service to listen on the incoming RPC. For server configurations, see section [2.1](#Section_e6076eab53f64aad9041888c5734b715).

### AD LDS Specifics

It is possible to run multiple [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) on the same computer. All of these AD LDS DCs listen on the same [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface ID. So that clients can distinguish between different instances of AD LDS that are running on the same computer, each RPC [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) is annotated (as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824)) with a string containing the [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) port number on which the DC listens. For example, if two AD LDS DCs are running on a computer, with one listening on port 389 and the other listening on port 50000, the RPC endpoints of the AD LDS DCs are annotated with "389" and "50000", respectively.

For a client to establish an RPC connection to an AD LDS DC, the client needs to know the name of the computer and the number of the LDAP port on which the AD LDS DC is listening. First, the client establishes a connection to the endpoint mapper service on the computer. Next, the client enumerates all endpoints that are registered for the desired interface ID. Finally, the client selects the endpoint whose annotation equals the LDAP port number of the desired AD LDS DC.

[**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024) DCs do not annotate their RPC endpoints. RPC endpoint annotation is not required for AD DS, because it is not possible to run multiple AD DS DCs on a computer.

# RPC Methods and Their Behavior

The methods for the drsuapi [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface are described in section [4.1](#Section_58f33216d9f143bfa18387e3c899c410).

The methods for the dsaop RPC interface are described in section [4.2](#Section_8988d95c631b46a4b84e16de204fb142).

## drsuapi RPC Interface

This section specifies the methods for the drsuapi [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface of this protocol and the processing rules for the methods.[<5>](#Appendix_A_5" \o "Product behavior note 5)

Methods in RPC [**Opnum**](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7) Order

| Method | Description |
| --- | --- |
| [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) | Creates a context handle necessary to call any other method in this interface.  Opnum: 0 |
| [IDL\_DRSUnbind](#Section_49eb17c9b6a94ceabef866abda8a7850) | Destroys a context handle previously created by the IDL\_DRSBind method.  Opnum: 1 |
| [IDL\_DRSReplicaSync](#Section_25c71d91051f4c26977fa70892f29b00) | Triggers [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) from another [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).  Opnum: 2 |
| [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) | Replicates [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) from an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) on the server.  Opnum: 3 |
| [IDL\_DRSUpdateRefs](#Section_a273bbcfaeca46088ad4127d3e597cd4) | Adds or deletes a value from the repsTo [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of a specified NC replica.  Opnum: 4 |
| [IDL\_DRSReplicaAdd](#Section_7219df914eea494f88e3780d40d2d559) | Adds a replication source reference for the specified [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942).  Opnum: 5 |
| [IDL\_DRSReplicaDel](#Section_1420a9bf9267464da6d57676472d7f1d) | Deletes a replication source reference for the specified NC.  Opnum: 6 |
| [IDL\_DRSReplicaModify](#Section_cd241bf256be453786b1cdbc997b0860) | Updates the value for repsFrom for the NC replica.  Opnum: 7 |
| [IDL\_DRSVerifyNames](#Section_80739a29e8ed44788490475a18e9779d) | Resolves a sequence of [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) identities.  Opnum: 8 |
| [IDL\_DRSGetMemberships](#Section_d5ace4527cdd4d50bb6439b4c55180a2) | Retrieves [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) membership for an object.  Opnum: 9 |
| [IDL\_DRSInterDomainMove](#Section_595b2fef493b4b1db60de7a1a3345b0e) | A helper method used in a cross-NC move [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) operation.  Opnum: 10 |
| [IDL\_DRSGetNT4ChangeLog](#Section_6e000eb660fd4d6cae82bb6479df02fa) | Returns a sequence of [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) change log entries or the Windows NT 4.0 operating system replication state.  Opnum: 11 |
| [IDL\_DRSCrackNames](#Section_9b4bfb4466564404bcc8dc88111658b3) | Looks up each of a set of objects in the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) and returns it to the caller in the requested format.  Opnum: 12 |
| [IDL\_DRSWriteSPN](#Section_8b129dc8ed4545379555b6fef764ab7d) | Updates the set of [**service principal names (SPNs)**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) on an object.  Opnum: 13 |
| [IDL\_DRSRemoveDsServer](#Section_d5c310ae347a49d4a78e6ffb2eecd581) | Removes the representation of a DC from the directory.  Opnum: 14 |
| [IDL\_DRSRemoveDsDomain](#Section_aa3cfa46c737425aae65ecaf9efe7e84) | Removes the representation of a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) from the directory.  Opnum: 15 |
| [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) | Retrieves information about DCs in a given domain.  Opnum: 16 |
| [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) | Adds one or more objects.  Opnum: 17 |
| [IDL\_DRSExecuteKCC](#Section_ad807917687b40d9abe2053af0246523) | Validates the replication interconnections of DCs and updates them if necessary.  Opnum: 18 |
| [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47) | Retrieves the replication state of the server.  Opnum: 19 |
| [IDL\_DRSAddSidHistory](#Section_376230a5d8064ae5970af6243ee193c8) | Adds one or more [**SIDs**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) to the sIDHistory attribute of a given object.  Opnum: 20 |
| [IDL\_DRSGetMemberships2](#Section_d4e67cc32ee14b2b8055cebefc556252) | Retrieves group memberships for a sequence of objects.  Opnum: 21 |
| [IDL\_DRSReplicaVerifyObjects](#Section_8dba150d50f647f1941e1a606c30db0b) | Verifies the existence of objects in an NC replica.  Opnum: 22 |
| [IDL\_DRSGetObjectExistence](#Section_6355d4f5f5564527adde37afba2fcf56) | Helps the client check the consistency of object existence between its [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) of an NC and the server's replica of the same NC.  Opnum: 23 |
| [IDL\_DRSQuerySitesByCost](#Section_2c3faba2d64e4866b8f1fc8d5f4ec710) | Determines the communication cost from a "from" [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba) to one or more "to" sites.  Opnum: 24 |
| [IDL\_DRSInitDemotion](#Section_faca17da3f7f409298dbfd2ce7d98b8c) | Performs the first phase of the removal of a DC from an [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62).  Opnum: 25 |
| [IDL\_DRSReplicaDemotion](#Section_8a2f0388bdfb4519a8c3384f27c11639) | Replicates off all changes to the specified NC and moves any [**FSMOs**](#gt_3fcc9e5e-60b6-40f8-acb6-ad3189cf90ec) held to another server.  Opnum: 26 |
| [IDL\_DRSFinishDemotion](#Section_0bf530e81be04f48b8c2208031a8725f) | Finishes or cancels the removal of a DC from an AD LDS forest.  Opnum: 27 |
| [IDL\_DRSAddCloneDC](#Section_ef0bfb1d037b4626a6d9cc7589bc5786) | Creates a new domain controller object by copying attributes from an existing domain controller object.  Opnum: 28 |
| [IDL\_DRSWriteNgcKey](#Section_7a140389caa34718bb1ad64483933eb0) | Composes and updates the msDS-KeyCredentialLink value on an object.  Opnum: 29 |
| [IDL\_DRSReadNgcKey](#Section_a80c60ac9864444a95136c0c894fbb8d) | Reads and parses the msDS-KeyCredentialLink value on an object.  Opnum: 30 |

The methods will affect only the directory instance that is bound to the current context. If a server has several directory instances installed, the other instances will remain unchanged.

The following considerations apply to the order of method calls. See section [1.3.2](#Section_67c5a415a6c740988cf36ef8d173cfe8) for details.

* IDL\_DRSBind must be called before any other method in order to obtain a context handle.
* After the IDL\_DRSUnbind method is called, the context handle that was passed to IDL\_DRSUnbind cannot be used for other method calls.
* IDL\_DRSInitDemotion is called before the other demotion methods.
* All other method calls are independent, apart from their dependencies on the state of the directory.

Because the order of method call is generally nonsequential (except as noted above), the method sections following this section are arranged alphabetically by method name.

### IDL\_DRSAddEntry (Opnum 17)

The IDL\_DRSAddEntry method adds one or more [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

1. ULONG IDL\_DRSAddEntry(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_ADDENTRYREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_ADDENTRYREPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle that is returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_ADDENTRYREQ

The DRS\_MSG\_ADDENTRYREQ union defines the request messages that are sent to the [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_ADDENTRYREQ\_V1 V1;
6. [case(2)]
7. DRS\_MSG\_ADDENTRYREQ\_V2 V2;
8. [case(3)]
9. DRS\_MSG\_ADDENTRYREQ\_V3 V3;
10. } DRS\_MSG\_ADDENTRYREQ;

**V1:**  Version 1 request (obsolete).

**V2:**  Version 2 request.

**V3:**  Version 3 request.

##### DRS\_MSG\_ADDENTRYREQ\_V1

The DRS\_MSG\_ADDENTRYREQ\_V1 structure defines the request message sent to the [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) method. This request version is obsolete.[<6>](#Appendix_A_6" \o "Product behavior note 6)

1. typedef struct {
2. [ref] DSNAME\* pObject;
3. ATTRBLOCK AttrBlock;
4. } DRS\_MSG\_ADDENTRYREQ\_V1;

**pObject:**  The identity of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to add.

**AttrBlock:**  The [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of the object to add.

##### DRS\_MSG\_ADDENTRYREQ\_V2

The DRS\_MSG\_ADDENTRYREQ\_V2 structure defines the request message sent to the [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) method.

1. typedef struct {
2. ENTINFLIST EntInfList;
3. } DRS\_MSG\_ADDENTRYREQ\_V2;

**EntInfList:**  The [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to be added, as specified in section [5.57](#Section_59cad1ffe499477c8c9e59939a71aff5).

##### DRS\_MSG\_ADDENTRYREQ\_V3

The DRS\_MSG\_ADDENTRYREQ\_V3 structure defines the request message sent to the [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) method.

1. typedef struct {
2. ENTINFLIST EntInfList;
3. DRS\_SecBufferDesc\* pClientCreds;
4. } DRS\_MSG\_ADDENTRYREQ\_V3;

**EntInfList:**  The [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to be added.

**pClientCreds:**  The user credentials to authorize the operation.

##### DRS\_MSG\_ADDENTRYREPLY

The DRS\_MSG\_ADDENTRYREPLY union defines the response messages received from the [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_ADDENTRYREPLY\_V1 V1;
6. [case(2)]
7. DRS\_MSG\_ADDENTRYREPLY\_V2 V2;
8. [case(3)]
9. DRS\_MSG\_ADDENTRYREPLY\_V3 V3;
10. } DRS\_MSG\_ADDENTRYREPLY;

**V1:**  Version 1 response (obsolete).

**V2:**  Version 2 response.

**V3:**  Version 3 response.

##### DRS\_MSG\_ADDENTRYREPLY\_V1

The DRS\_MSG\_ADDENTRYREPLY\_V1 structure defines the response message received from the [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) method. This response version is obsolete.[<7>](#Appendix_A_7" \o "Product behavior note 7)

1. typedef struct {
2. GUID Guid;
3. NT4SID Sid;
4. DWORD errCode;
5. DWORD dsid;
6. DWORD extendedErr;
7. DWORD extendedData;
8. USHORT problem;
9. } DRS\_MSG\_ADDENTRYREPLY\_V1;

**Guid:**   The objectGUID of the added [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

**Sid:**   The objectSid of the added object.

**errCode:**  0 if successful or a DIRERR error code (section [4.1.1.1.25](#Section_b5165b5e399a47d685d345db0296acbb)) if a fatal error occurred.

**dsid:**  The implementation-specific diagnostic code.

**extendedErr:**  0, [**STATUS code**](#gt_dfc7ec7a-2b99-4312-a947-5d90a117d1c7), or [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**extendedData:**  The implementation-specific diagnostic code.

**problem:**  0 or PROBLEM error code (section [4.1.1.1.26](#Section_4e239b71e4544908a7591fa412312db7)).

##### DRS\_MSG\_ADDENTRYREPLY\_V2

The DRS\_MSG\_ADDENTRYREPLY\_V2 structure defines the response message received from the [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) method.

1. typedef struct {
2. [unique] DSNAME\* pErrorObject;
3. DWORD errCode;
4. DWORD dsid;
5. DWORD extendedErr;
6. DWORD extendedData;
7. USHORT problem;
8. [range(0,10000)] ULONG cObjectsAdded;
9. [size\_is(cObjectsAdded)] ADDENTRY\_REPLY\_INFO\* infoList;
10. } DRS\_MSG\_ADDENTRYREPLY\_V2;

**pErrorObject:**  Null, or the identity of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that was being added when an error occurred.

**errCode:**  0 if successful, otherwise a DIRERR error code (section [4.1.1.1.25](#Section_b5165b5e399a47d685d345db0296acbb)).

**dsid:**  The implementation-specific diagnostic code.

**extendedErr:**  0, [**STATUS code**](#gt_dfc7ec7a-2b99-4312-a947-5d90a117d1c7), or [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**extendedData:**  The implementation-specific diagnostic code.

**problem:**  0 or PROBLEM error code (section [4.1.1.1.26](#Section_4e239b71e4544908a7591fa412312db7)).

**cObjectsAdded:**  The count of items in the *infoList* array.

**infoList:**  The identities of the added objects. The item order matches the item order of values in the **EntInfList** field in the request structure.

##### DRS\_MSG\_ADDENTRYREPLY\_V3

The DRS\_MSG\_ADDENTRYREPLY\_V3 structure defines the response message received from the [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) method.

1. typedef struct {
2. DSNAME\* pdsErrObject;
3. DWORD dwErrVer;
4. [switch\_is(dwErrVer)] DRS\_ERROR\_DATA\* pErrData;
5. [range(0,10000)] ULONG cObjectsAdded;
6. [size\_is(cObjectsAdded)] ADDENTRY\_REPLY\_INFO\* infoList;
7. } DRS\_MSG\_ADDENTRYREPLY\_V3;

**pdsErrObject:**  Null, or the identity of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that was being added when an error occurred.

**dwErrVer:**  MUST be set to 1.

**pErrData:**  Null, or an error that occurred while processing the request.

**cObjectsAdded:**  The count of items in the **infoList** array.

**infoList:**  The identities of the added objects. The item order matches the item order of values in the **EntInfList** field in the request structure.

##### ADDENTRY\_REPLY\_INFO

The ADDENTRY\_REPLY\_INFO structure defines the identity of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) added by the [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) method.

1. typedef struct {
2. GUID objGuid;
3. NT4SID objSid;
4. } ADDENTRY\_REPLY\_INFO;

**objGuid:**  The objectGUID of the added object.

**objSid:**  The objectSid of the added object.

##### DIRERR\_DRS\_WIRE\_V1

The DIRERR\_DRS\_WIRE\_V1 union defines the error that occurred during processing of a request sent to the [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. ATRERR\_DRS\_WIRE\_V1 AtrErr;
6. [case(2)]
7. NAMERR\_DRS\_WIRE\_V1 NamErr;
8. [case(3)]
9. REFERR\_DRS\_WIRE\_V1 RefErr;
10. [case(4)]
11. SECERR\_DRS\_WIRE\_V1 SecErr;
12. [case(5)]
13. SVCERR\_DRS\_WIRE\_V1 SvcErr;
14. [case(6)]
15. UPDERR\_DRS\_WIRE\_V1 UpdErr;
16. [case(7)]
17. SYSERR\_DRS\_WIRE\_V1 SysErr;
18. } DIRERR\_DRS\_WIRE\_V1;

**AtrErr:**  [**Attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) errors.

**NamErr:**  Name resolution error.

**RefErr:**  Referral.

**SecErr:**  Security error.

**SvcErr:**  Service error.

**UpdErr:**  Update error.

**SysErr:**  System error.

##### ATRERR\_DRS\_WIRE\_V1

The ATRERR\_DRS\_WIRE\_V1 structure defines [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) errors.

1. typedef struct {
2. DSNAME\* pObject;
3. ULONG count;
4. PROBLEMLIST\_DRS\_WIRE\_V1 FirstProblem;
5. } ATRERR\_DRS\_WIRE\_V1;

**pObject:**  The identity of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) being processed when the error occurred.

**count:**  The count of items in the **FirstProblem** linked list.

**FirstProblem:**  The first element in the linked list of attribute errors.

##### PROBLEMLIST\_DRS\_WIRE\_V1

The PROBLEMLIST\_DRS\_WIRE\_V1 structure defines an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) error link entry.

1. typedef struct \_PROBLEMLIST\_DRS\_WIRE\_V1 {
2. struct \_PROBLEMLIST\_DRS\_WIRE\_V1\* pNextProblem;
3. INTFORMPROB\_DRS\_WIRE\_V1 intprob;
4. } PROBLEMLIST\_DRS\_WIRE\_V1;

**pNextProblem:**  Null, or a pointer to the next item in the list.

**intprob:**  Attribute error description.

##### INTFORMPROB\_DRS\_WIRE\_V1

The INTFORMPROB\_DRS\_WIRE\_V1 structure defines an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) error.

1. typedef struct {
2. DWORD dsid;
3. DWORD extendedErr;
4. DWORD extendedData;
5. USHORT problem;
6. ATTRTYP type;
7. BOOL valReturned;
8. ATTRVAL Val;
9. } INTFORMPROB\_DRS\_WIRE\_V1;

**dsid:**  The implementation-specific diagnostic code.

**extendedErr:**  0, [**STATUS code**](#gt_dfc7ec7a-2b99-4312-a947-5d90a117d1c7), or [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**extendedData:**  The implementation-specific diagnostic code.

**problem:**  0 or PROBLEM error code (section [4.1.1.1.26](#Section_4e239b71e4544908a7591fa412312db7)).

**type:**  The attribute that was being processed when the error occurred.

**valReturned:**  If true, the offending value is returned in the Val member.

**Val:**  The offending value.

##### NAMERR\_DRS\_WIRE\_V1

The NAMERR\_DRS\_WIRE\_V1 structure defines a name resolution error.

1. typedef struct {
2. DWORD dsid;
3. DWORD extendedErr;
4. DWORD extendedData;
5. USHORT problem;
6. DSNAME\* pMatched;
7. } NAMERR\_DRS\_WIRE\_V1;

**dsid:**  The implementation-specific diagnostic code.

**extendedErr:**  0, [**STATUS code**](#gt_dfc7ec7a-2b99-4312-a947-5d90a117d1c7), or [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**extendedData:**  The implementation-specific diagnostic code.

**problem:**  0 or PROBLEM error code (section [4.1.1.1.26](#Section_4e239b71e4544908a7591fa412312db7)).

**pMatched:**  The best match for the supplied [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) identity.

##### REFERR\_DRS\_WIRE\_V1

The REFERR\_DRS\_WIRE\_V1 structure defines a referral to other [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

1. typedef struct {
2. DWORD dsid;
3. DWORD extendedErr;
4. DWORD extendedData;
5. CONTREF\_DRS\_WIRE\_V1 Refer;
6. } REFERR\_DRS\_WIRE\_V1;

**dsid:**  The implementation-specific diagnostic code.

**extendedErr:**  0, [**STATUS code**](#gt_dfc7ec7a-2b99-4312-a947-5d90a117d1c7), or [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**extendedData:**  The implementation-specific diagnostic code.

**Refer:**  The DCs to contact to chase the referral.

##### NAMERESOP\_DRS\_WIRE\_V1

The NAMERESOP\_DRS\_WIRE\_V1 structure defines the state of name resolution.

1. typedef struct {
2. UCHAR nameRes;
3. UCHAR unusedPad;
4. USHORT nextRDN;
5. } NAMERESOP\_DRS\_WIRE\_V1;

**nameRes:**  MUST be the uppercase ASCII character "S".

**unusedPad:**  Unused. MUST be 0 and ignored.

**nextRDN:**  Unused. MUST be 0 and ignored.

##### DSA\_ADDRESS\_LIST\_DRS\_WIRE\_V1

The DSA\_ADDRESS\_LIST\_DRS\_WIRE\_V1 structure defines a linked list entry for a referral network name.

1. typedef struct \_DSA\_ADDRESS\_LIST\_DRS\_WIRE\_V1 {
2. struct \_DSA\_ADDRESS\_LIST\_DRS\_WIRE\_V1\* pNextAddress;
3. RPC\_UNICODE\_STRING\* pAddress;
4. } DSA\_ADDRESS\_LIST\_DRS\_WIRE\_V1;

**pNextAddress:**  Null, or the next element in the linked list.

**pAddress:**  Network name of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) to which the referral is directed.

##### CONTREF\_DRS\_WIRE\_V1

The CONTREF\_DRS\_WIRE\_V1 structure defines a linked list entry for a continuation referral.

1. typedef struct CONTREF\_DRS\_WIRE\_V1 {
2. DSNAME\* pTarget;
3. NAMERESOP\_DRS\_WIRE\_V1 OpState;
4. USHORT aliasRDN;
5. USHORT RDNsInternal;
6. USHORT refType;
7. USHORT count;
8. DSA\_ADDRESS\_LIST\_DRS\_WIRE\_V1\* pDAL;
9. struct CONTREF\_DRS\_WIRE\_V1\* pNextContRef;
10. BOOL bNewChoice;
11. UCHAR choice;
12. } CONTREF\_DRS\_WIRE\_V1;

**pTarget:**  The [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to which the referral is directed.

**OpState:**  The operation state.

**aliasRDN:**  Unused. MUST be 0 and ignored.

**RDNsInternal:**  Unused. MUST be 0 and ignored.

**refType:**  The type of referral. This field MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| CH\_REFTYPE\_SUPERIOR  0x0000 | A referral to a superior [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). |
| CH\_REFTYPE\_SUBORDINATE  0x0001 | A referral to a subordinate DC (for example, to a child [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca)). |
| CH\_REFTYPE\_NSSR  0x0002 | Not in use. |
| CH\_REFTYPE\_CROSS  0x0003 | A referral to an external crossRef object. See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.1.2.1.1.1. |

**count:**  The count of items in the **pDAL** linked list.

**pDAL:**  A list of network names of the DCs to which the referral is directed.

**pNextContRef:**  Null, or the next item in the linked list.

**bNewChoice:**  True if and only if a new choice is specified.

**choice:**  The choice to use in the continuation referral. This field MUST be one of the following values:

| Value | Meaning |
| --- | --- |
| SE\_CHOICE\_BASE\_ONLY  0x00 | A base search is to be performed. |
| SE\_CHOICE\_IMMED\_CHLDRN  0x01 | A one-level search is to be performed. |
| SE\_CHOICE\_WHOLE\_SUBTREE  0x02 | A subtree search is to be performed. |

##### SECERR\_DRS\_WIRE\_V1

The SECERR\_DRS\_WIRE\_V1 structure defines a security error.

1. typedef struct {
2. DWORD dsid;
3. DWORD extendedErr;
4. DWORD extendedData;
5. USHORT problem;
6. } SECERR\_DRS\_WIRE\_V1;

**dsid:**  The implementation-specific diagnostic code.

**extendedErr:**  0, [**STATUS code**](#gt_dfc7ec7a-2b99-4312-a947-5d90a117d1c7), or [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**extendedData:**  The implementation-specific diagnostic code.

**problem:**  0 or PROBLEM error code (section [4.1.1.1.26](#Section_4e239b71e4544908a7591fa412312db7)).

##### SVCERR\_DRS\_WIRE\_V1

The SVCERR\_DRS\_WIRE\_V1 structure defines a service error.

1. typedef struct {
2. DWORD dsid;
3. DWORD extendedErr;
4. DWORD extendedData;
5. USHORT problem;
6. } SVCERR\_DRS\_WIRE\_V1;

**dsid:**  The implementation-specific diagnostic code.

**extendedErr:**  0, [**STATUS code**](#gt_dfc7ec7a-2b99-4312-a947-5d90a117d1c7), or [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**extendedData:**  The implementation-specific diagnostic code.

**problem:**  0 or PROBLEM error code (section [4.1.1.1.26](#Section_4e239b71e4544908a7591fa412312db7)).

##### UPDERR\_DRS\_WIRE\_V1

The UPDERR\_DRS\_WIRE\_V1 structure defines an [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) error.

1. typedef struct {
2. DWORD dsid;
3. DWORD extendedErr;
4. DWORD extendedData;
5. USHORT problem;
6. } UPDERR\_DRS\_WIRE\_V1;

**dsid:**  The implementation-specific diagnostic code.

**extendedErr:**  0, [**STATUS code**](#gt_dfc7ec7a-2b99-4312-a947-5d90a117d1c7), or [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**extendedData:**  The implementation-specific diagnostic code.

**problem:**  0 or PROBLEM error code (section [4.1.1.1.26](#Section_4e239b71e4544908a7591fa412312db7)).

##### SYSERR\_DRS\_WIRE\_V1

The SYSERR\_DRS\_WIRE\_V1 structure defines a system error.

1. typedef struct {
2. DWORD dsid;
3. DWORD extendedErr;
4. DWORD extendedData;
5. USHORT problem;
6. } SYSERR\_DRS\_WIRE\_V1;

**dsid:**  The implementation-specific diagnostic code.

**extendedErr:**  0, [**STATUS code**](#gt_dfc7ec7a-2b99-4312-a947-5d90a117d1c7), or [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**extendedData:**  The implementation-specific diagnostic code.

**problem:**  0 or PROBLEM error code (section [4.1.1.1.26](#Section_4e239b71e4544908a7591fa412312db7)).

##### DRS\_ERROR\_DATA

The DRS\_ERROR\_DATA union defines the error responses that are received from the [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_ERROR\_DATA\_V1 V1;
6. } DRS\_ERROR\_DATA;

**V1:**  Version 1 response.

##### DRS\_ERROR\_DATA\_V1

The DRS\_ERROR\_DATA\_V1 structure defines the error response received from the [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) method.

1. typedef struct {
2. DWORD dwRepError;
3. DWORD errCode;
4. [switch\_is(errCode)] DIRERR\_DRS\_WIRE\_V1\* pErrInfo;
5. } DRS\_ERROR\_DATA\_V1;

**dwRepError:**  0 or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**errCode:**  A DIRERR code (section [4.1.1.1.25](#Section_b5165b5e399a47d685d345db0296acbb)) that specifies the error category.

**pErrInfo:**  Category-specific error information.

##### DIRERR Codes

The DIRERR codes classify an error that occurs during a search for, or the [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) of, a [**directory object**](#gt_407dbc2c-3140-4e31-9085-0087e2d3bab2).

| Value and symbolic name | Description |
| --- | --- |
| 0x00000001  attributeError | [**Attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) errors. |
| 0x00000002  nameError | Name resolution error. |
| 0x00000003  referralError | Referral. |
| 0x00000004  securityError | Security error. |
| 0x00000005  serviceError | Service error. |
| 0x00000006  updError | Update error. |
| 0x00000007  systemError | System error. |

##### PROBLEM Error Codes

The PROBLEM error codes describe the problems that can be reported by an [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) operation.

| Value and symbolic name | Description |
| --- | --- |
| 0x0000000C  ENOMEM | Out of memory. |
| 0x00000010  EBUSY | Too busy to proceed. |
| 0x00000016  EINVAL | Invalid parameter. |
| 0x0000001C  ENOSPC | Out of storage. |
| 0x000007D1  NA\_PROBLEM\_NO\_OBJECT | [**Parent object**](#gt_0d41951a-62f0-4fbd-bb23-22f645ae3bf5) not found in the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). |
| 0x000003E9  PR\_PROBLEM\_NO\_ATTRIBUTE\_OR\_VAL | [**Attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) or value not found. |
| 0x000003EA  PR\_PROBLEM\_INVALID\_ATT\_SYNTAX | Invalid attribute syntax. |
| 0x000003EB  PR\_PROBLEM\_UNDEFINED\_ATT\_TYPE | Unknown attribute type. |
| 0x000003EC  PR\_PROBLEM\_WRONG\_MATCH\_OPER | Incorrect matching operation (only applies to match operators in [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) filters). |
| 0x000003ED  PR\_PROBLEM\_CONSTRAINT\_ATT\_TYPE | Attribute value violated a [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093) constraint. |
| 0x000003EE  PR\_PROBLEM\_ATT\_OR\_VALUE\_EXISTS | Attribute or value already exists (multiple values specified for a single-valued attribute OR duplicate value specified for a multi-valued attribute). |
| 0x00000FA1  SE\_PROBLEM\_INAPPROPRIATE\_AUTH | Inappropriate [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) method. |
| 0x00000FA2  SE\_PROBLEM\_INVALID\_CREDENTS | Invalid user name or password. |
| 0x00000FA3  SE\_PROBLEM\_INSUFF\_ACCESS\_RIGHTS | Access denied. |
| 0x00000FA4  SE\_PROBLEM\_INVALID\_SIGNATURE | Invalid signature. |
| 0x00000FA5  SE\_PROBLEM\_PROTECTION\_REQUIRED | Encrypted connection required. |
| 0x00000FA6  SE\_PROBLEM\_NO\_INFORMATION | Insufficient permissions to generate a referral. |
| 0x00001389  SV\_PROBLEM\_BUSY | [**Directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) service is busy. |
| 0x0000138A  SV\_PROBLEM\_UNAVAILABLE | Directory service is unavailable. |
| 0x0000138B  SV\_PROBLEM\_WILL\_NOT\_PERFORM | The requested operation will not be performed. |
| 0x0000138C  SV\_PROBLEM\_CHAINING\_REQUIRED | Chaining is required to perform the operation. |
| 0x0000138D  SV\_PROBLEM\_UNABLE\_TO\_PROCEED | Directory service is unable to proceed with the requested operation. |
| 0x0000138E  SV\_PROBLEM\_INVALID\_REFERENCE | Invalid crossRef [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). |
| 0x0000138F  SV\_PROBLEM\_TIME\_EXCEEDED | Time limit exceeded while processing the operation. |
| 0x00001390  SV\_PROBLEM\_ADMIN\_LIMIT\_EXCEEDED | Administrative limit exceeded while processing the operation. |
| 0x00001391  SV\_PROBLEM\_LOOP\_DETECTED | Not in use. |
| 0x00001392  SV\_PROBLEM\_UNAVAIL\_EXTENSION | The requested [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6) is not available. |
| 0x00001393  SV\_PROBLEM\_OUT\_OF\_SCOPE | Not in use. |
| 0x00001394  SV\_PROBLEM\_DIR\_ERROR | Generic directory service error. |
| 0x00001771  UP\_PROBLEM\_NAME\_VIOLATION | Naming violation. |
| 0x00001772  UP\_PROBLEM\_OBJ\_CLASS\_VIOLATION | [**Object class**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a) violation. |
| 0x00001773  UP\_PROBLEM\_CANT\_ON\_NON\_LEAF | The operation cannot be performed on an object with [**child objects**](#gt_9b04b599-9dca-48f1-aa9e-08e254d20553). |
| 0x00001774  UP\_PROBLEM\_CANT\_ON\_RDN | The operation cannot be performed on an [**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) attribute. |
| 0x00001775  UP\_PROBLEM\_ENTRY\_EXISTS | Object already exists. |
| 0x00001776  UP\_PROBLEM\_AFFECTS\_MULT\_DSAS | The operation affects multiple [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). |
| 0x00001777  UP\_PROBLEM\_CANT\_MOD\_OBJ\_CLASS | The [**objectClass**](#gt_4191a0f6-e528-4927-bf0e-7a9981e014c8) attribute cannot be modified in this way. |

#### Method-Specific Abstract Types and Procedures

##### ConstructReplSpn

1. procedure ConstructReplSpn(
2. dnsHostName: unicodestring,
3. guid: GUID): unicodestring

This procedure returns a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) with the given DNS host name (in *dnsHostName*) and [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) (in *guid*). The [**service class**](#gt_c1386afd-9d42-47c7-a7ea-d01912a17647) of the resulting SPN is [DRS\_SPN\_CLASS](#Section_3e0e2ef55dc54b37a927323f3c4876ca). For example, given *dnsHostName* = "dc-01.fabrikam.com" and *guid* being the GUID whose string representation is "{d66e9688-66a5-4a52-8af2-17b110febe0c}", the return value is:

E3514235-4B06-11D1-AB04-00C04FC2DCD2/d66e9688-66a5-4a52-8af2-17b110febe0c/dc-01.fabrikam.com

##### CreateCrossRef

1. procedure CreateCrossRef(
2. hDrs: DRS\_HANDLE,
3. e: ENTINF,
4. pmsgOut: ADDRESS OF DRS\_MSG\_ADDENTRYREPLY,
5. ver: DWORD,
6. info: ADDRESS OF ADDENTRY\_REPLY\_INFO): boolean

*Informative summary of behavior*: This procedure creates a crossRef [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). If the crossRef object exists already in a disabled state, it will mark the crossRef object as enabled.

1. ulSysFlags, err: DWORD
2. ncNameV: DSName
3. trustParentV, rootTrustV, dnsRootV: unicodestring
4. cr: DSName
5. prefixTable: PrefixTable
6. /\* Only attributes and classes in the base schema can be specified.\*/
7. prefixTable := NewPrefixTable()
8. ulSysFlags := ENTINF\_GetValue(e, systemFlags, prefixTable)
9. ncNameV := ENTINF\_GetValue(e, ncName, prefixTable)
10. /\* Check whether the crossRef object for the given ncName exists. \*/
11. cr := select one v from subtree ConfigNC()
12. where v!ncName = ncNameV and crossRef in v!objectClass
13. if (cr = null) or not (FLAG\_CR\_NTDS\_DOMAIN in ulSysFlags) then
14. if FLAG\_CR\_NTDS\_NC in ulSysFlags then
15. SetErrorData(SV\_PROBLEM\_WILL\_NOT\_PERFORM, serviceError,
16. ERROR\_DS\_MISSING\_EXPECTED\_ATT, pmsgOut, ver)
17. return false
18. endif
19. /\* Add the crossRef object as a regular operation; this is subject
20. \* to an access check and will succeed only if the server is the
21. \* Partition Naming Master FSMO role owner. \*/
22. err := PerformAddOperation(e, cr, dc.prefixTable, TRUE)
23. if err ≠ 0 then
24. /\* Pick up the error information from the previous call. \*/
25. SetErrorData(0, 0, 0, pmsgOut, ver)
26. return false
27. endif
28. /\* Set the systemFlags because PerformAddOperation does not set it.
29. \*/
30. cr!systemFlags := ulSysFlags
31. /\* Return the objectGUID of the new crossRef object. \*/
32. info^.objGuid := cr.guid;
33. else
34. /\* crossRef already exists; enable it. \*/
35. /\* The crossRef is expected to be disabled. \*/
36. if cr!enabled = null or cr!enabled = true then
37. SetErrorData(SV\_PROBLEM\_DIR\_ERROR,
38. serviceError,
39. ERROR\_DUP\_DOMAINNAME,
40. pmsgOut, ver)
41. return false
42. endif
43. /\* Only allow certain client IP to make the change. \*/
44. if not (ClientIpMatch(hDrs, cr!dnsRoot)) then
45. SetErrorData(SE\_PROBLEM\_INAPPROPRIATE\_AUTH, securityError,
46. ERROR\_DS\_INTERNAL\_FAILURE, pmsgOut, ver)
47. return false
48. endif
50. /\* dnsRoot must be set in the given ENTINF. \*/
51. dnsRootV := ENTINF\_GetValue(e, dnsRoot, prefixTable)
52. if dnsRootV = null then
53. SetErrorData(PR\_PROBLEM\_NO\_ATTRIBUTE\_OR\_VAL, attributeError,
54. ERROR\_DS\_MISSING\_REQUIRED\_ATT, pmsgOut, ver)
55. return false
56. endif
57. cr!dnsRoot := dnsRootV
58. /\* Two more attributes can be set; the rest are ignored. \*/
59. trustParentV := ENTINF\_GetValue(e, trustParent, prefixTable)
60. if trustParentV ≠ null then
61. cr!trustParent := trustParentV
62. endif
63. rootTrustV := ENTINF\_GetValue(e, rootTrust, prefixTable)
64. if rootTrustV ≠ null then
65. cr!rootTrust := rootTrustV
66. endif
67. /\* Update the systemFlags and enable the crossRef. \*/
68. cr!systemFlags := {FLAG\_CR\_NTDS\_NC, FLAG\_CR\_NTDS\_DOMAIN}
69. cr!enabled := null
70. /\* return the guid of the crossRef object \*/
71. info^.objGuid := cr.guid
72. endif
73. /\*The cross ref was created/enabled. Ensure that the respective
74. sub-ref object is created \*/
75. AddSubRef(cr!ncName)
76. return true

##### CreateNtdsDsa

1. procedure CreateNtdsDsa(
2. hDrs: DRS\_HANDLE,
3. e: ENTINF,
4. entList: ADDRESS OF ENTINFLIST,
5. pmsgOut: ADDRESS OF DRS\_MSG\_ADDENTRYREPLY,
6. ver: DWORD,
7. info: ADDRESS OF ADDENTRY\_REPLY\_INFO): boolean

*Informative summary of behavior*: This procedure creates an nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

1. domainName, domainCR, domain, cr, v,
2. partitionsObj, sl, dsaObj: DSName
3. accessAllowed: boolean
4. dcfl, err: DWORD
5. spn: unicodestring
6. prefixTable: PrefixTable
7. /\* Only attributes and classed in the base schema can be specified.\*/
8. prefixTable := NewPrefixTable()
9. domainName := GetDomainNameFromEntinf(e)
10. domainCR := select one v from ConfigNC() where v!nCName = domainName
11. and crossRef in v!objectClass
12. and FLAG\_CR\_NTDS\_DOMAIN in v!systemFlags
13. domain := select one v from all where v = domainName
14. if domain ≠ null then
15. /\* Perform access check. \*/
16. accessAllowed :=
17. AccessCheckCAR(domain, DS-Replication-Manage-Topology)
18. else
19. /\* Creating the domain crossRef in the same call is
20. \* allowed. The call will fail if the caller does not have right
21. \* to create the crossRef object. \*/
22. accessAllowed := IsDomainToBeCreated(entList, domain)
23. endif
24. if not accessAllowed then
25. SetErrorData(SV\_PROBLEM\_DIR\_ERROR, serviceError,
26. ERROR\_ACCESS\_DENIED, pmsgOut, ver)
27. return false
28. endif
29. /\* Check for the functional level compliance. The functional level
30. \* of a DC cannot be less than the functional level of the forest.
31. \* If the DC is not the first DC in is domain, its functional level
32. \* cannot be less than the functional level of its domain. \*/
33. dcfl := ENTINF\_GetValue(e, msDS-Behavior-Version, prefixTable)
34. if dcfl = null then
35. dcfl := 0
36. endif
37. if domain = DefaultNC() and
38. dcfl < DefaultNC()!msDS-Behavior-Version then
39. SetErrorData(SV\_PROBLEM\_WILL\_NOT\_PERFORM, serviceError,
40. ERROR\_DS\_INCOMPATIBLE\_VERSION, pmsgOut, ver)
41. return false
42. endif
43. partitionsObj := DescendantObject(ConfigNC(), "CN=Partitions,")
45. if dcfl < partitionsObj!msDS-Behavior-Version then
46. SetErrorData(SV\_PROBLEM\_WILL\_NOT\_PERFORM, serviceError,
47. ERROR\_DS\_INCOMPATIBLE\_VERSION, pmsgOut, ver)
48. return false
49. endif
50. /\* serverReference attribute is not updated here; instead, it is used
51. \* to find the computer object of the DC so that the replication SPN
52. \* can be added to the DC's computer object. \*/
53. sl := ENTINF\_GetValue(e, serverReference, prefixTable)
54. ENTINF\_SetValue(e, serverReference, null, prefixTable)
55. /\* Create the object in the system context; this is necessary to
56. \* avoid the system-only class constraint defined in the schema.\*/
57. err := PerformAddOperationAsSystem(e, dsaObj, prefixTable)
58. if err ≠ 0 then
59. /\* Pick up the error information PerformAddOperationAsSystem set.\*/
60. SetErrorData(0, 0, 0, pmsgOut, ver)
61. return false
62. endif
63. /\* Find the computer object and update its SPN. \*/
64. if sl ≠ null then
65. dcObj := select one v from subtree DefaultNC() where v = sl
66. spn := ConstructReplSpn(domainCR!dnsHostName, dcObj.guid)
67. dcObj!servicePrincipalName := dcObj!servicePrincipalName + {spn}
68. endif
69. /\* Return the objectGUID of the new nTDSDSA object. \*/
70. info^.objGuid := dsaObj.guid
71. return true

##### UseCredsForAccessCheck

1. procedure UseCredsForAccessCheck(creds: DRS\_SecBufferDesc): DWORD

This procedure gets authorization information for a client (using the [ClientAuthorizationInfo](#Section_92e47a548f244182b6b484f28699f8a1) [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b), which is a security token) by authenticating the given credentials. Any access checks performed during the remainder of the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) call are performed against this information.

##### IsDomainToBeCreated

1. procedure IsDomainToBeCreated(
2. entList: ADDRESS OF ENTINFLIST,
3. ncName: DSName): boolean

This procedure searches all the [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b) values in *entList* for any request to create a crossRef [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *cr* such that *cr*!nCName = *ncName*. It returns true if such a *cr* is found; otherwise, it returns false.

##### GetDomainNameFromEntinf

1. procedure GetDomainNameFromEntinf(e: ENTINF): DSName

*Informative summary of behavior*: This procedure examines the values for the hasMasterNCs [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) found in *e* and returns the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef). The hasMasterNCs attribute always contains the dsnames of the [**schema NC**](#gt_754c2f7e-1fe4-4d1b-8976-6dad8d13e450), the [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625), and the default domain NC of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) represented by *e*. The domain NC is identified by a process of elimination.

1. prefixTable: PrefixTable
2. attr:ATTR
3. j:DWORD
4. prefixTable := NewPrefixTable()
5. /\* Scan the ENTINF e to get the attribute for which ATTRTYP is
6. \* hasMasterNCs.\*/
7. attr := ENTINF\_GetAttribute(e, hasMasterNCs, prefixTable)
8. for j=0 to (attr.AttrVal.valCount-1)
9. if (attr.AttrVal.pAVal[j].pVal ≠ SchemaNC()) and
10. (attr.AttrVal.pAVal[j].pVal ≠ ConfigNC()) and
11. (attr.AttrVal.pAVal[j].valLen > 0) then
12. return attr.AttrVal.pAVal[j].pVal^
13. end if
14. endfor
15. return null

##### ENTINF\_GetAttribute

1. procedure ENTINF\_GetAttribute (
2. entInf: ENTINF,
3. attribute: ATTRTYP,
4. prefixTable: PrefixTable
5. ): ATTR

*Informative summary of behavior*: The ENTINF\_GetAttribute procedure scans an [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b) structure and returns the first [ATTR](#Section_a2db41e278034d3ca4990fee92b1c149) structure for the requested *attribute*. The *attribute* parameter is based on [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).*prefixTable*, while the [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) within *entInf* are based on the *prefixTable* parameter.

1. attrType: ATTRTYP
2. oid : OID
3. oid := OidFromAttid(dc.prefixTable, attribute)
4. attrType := MakeAttid(prefixTable, oid)
5. for each i in [0 .. entInf.AttrBlock.attrCount-1] do
6. if (entInf.AttrBlock.pAttr[i].attrTyp = attrType) then
7. return entInf.AttrBlock.pAttr[i]
8. endif
9. endfor
10. return null

##### SetErrorData

1. procedure SetErrorData(
2. problem: USHORT,
3. errCode: ULONG,
4. extendedError: ULONG,
5. pmsgOut: ADDRESS OF DRS\_MSG\_ADDENTRYREPLY,
6. version: ULONG)

This procedure sets the error message fields of **pmsgOut**: the **problem**, **errCode**, and **extendedErr** fields of *pmsgOut^*.*V2* if *version* = 2 or the **pErrData** field of *pmsgOut^*.*V3* if *version* = 3. If **problem**, **errCode**, and **extendedError** are all 0, the error information is the result of the last call to [PerformAddOperation](#Section_e61c9ac3cd7c47d3ab1b0c8b61dc4869) or [PerformAddOperationAsSystem](#Section_2b75b1b502c942b48e926b154bd2893b).

##### ClientIpMatch

1. procedure ClientIpMatch(
2. hDrs: DRS\_HANDLE,
3. dnsRoot: set of unicodestring): boolean

This function returns true if the IP address of the client with [DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) *hDrs* matches one of the IP addresses of the DNS host names in the set *dnsRoot*.

##### PerformModifyEntInf

1. procedure PerformModifyEntInf(
2. hDrs: DRS\_HANDLE,
3. e: ENTINF,
4. info: ADDRESS OF ADDENTRY\_REPLY\_INFO): boolean

This function performs a modify operation on the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *e.*pName^. It enforces all security, [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093), and other constraints and follows all processing rules as used by the [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) modify operation (see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a)). The objectGUID and objectSid of the object being modified are returned in the info output structure. If the operation succeeds, PerformModifyEntInf returns true. If the operation fails for some reason, PerformModifyEntInf sets an appropriate error code (as defined by the LDAP modify operation) in the *info* structure, and returns false.

#### Server Behavior of the IDL\_DRSAddEntry Method

*Informative summary of behavior*: A *disabled* crossRef [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *cr* is one with *cr*!Enabled = false. *Enabling* a disabled crossRef object *cr* means setting *cr*!nCName and *cr*!dnsRoot, and removing *cr*!Enabled.

This method enables, creates, or modifies one or more objects, as requested by the client, in a single transaction. It enables crossRef objects, creates crossRef objects and nTDSDSA objects, and modifies arbitrary objects. The client uses an [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b) structure to specify the state of each enabled, created, or modified object:

* Enabling a crossRef object: The dnsRoot [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of a disabled crossRef object contains a set of one or more DNS host names, expressed as [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) strings. The request to enable a crossRef object succeeds only if the IP address of the client that is making the request matches the IP address of one of the DNS host names in the dnsRoot attribute. When a disabled crossRef object is enabled through this method, the server is not required to be the Domain Naming Master [**FSMO role owner**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b).

The client has to specify the nCName and dnsRoot attributes. The trustParent and rootTrust attributes are optional.

* Creating a crossRef object: If the request creates a crossRef object, it succeeds only if the server owns the [**forest's**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) Domain Naming Master [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f). The access check is the same as when a crossRef object is created through [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d).

The client specifies the same attributes that are required during an LDAP Add of a crossRef object, namely the new object's [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b), plus all must-have attributes of the crossRef [**class**](#gt_18393bbe-0c06-42b7-890d-b94a9a40b6e0). See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.1.2.1.1 for the specification of crossRef objects.

* Creating an nTDSDSA object: Creating an nTDSDSA object is not possible with LDAP. To create an nTDSDSA object, the hasMasterNCs attribute in the request has to identify the forest's [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093) [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) and [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625), and the [**DC's**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) [**default NC**](#gt_adc2c434-9679-419f-8c8b-b8c234921ad3); that is, the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) of the DC corresponding to the new nTDSDSA object. If the default NC exists on the server as the nTDSDSA object is being created by [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e), the client has to have the [**control access right**](#gt_42f6c9e0-a2b3-4bc3-9b87-fdb902e5505e) DS-Replication-Manage-Topology on the default NC. Otherwise, the client has to have the right to enable or create the crossRef object that corresponds to the default NC, and has to enable or create this crossRef object in the same IDL\_DRSAddEntry request.

The client specifies the new object's DN, plus the hasMasterNCs attribute. To create an nTDSDSA object for a functional DC, the request will contain invocationId, dMDLocation, options, msDS-Behavior-Version, and systemFlags. See [MS-ADTS] section 6.1.1.2.2.1.2.1.1 for the specification of nTDSDSA objects.

If the serverReference attribute is given a value in the request, the computer object to which the serverReference attribute points is [**updated**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) with a new [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4).

* Modifying an object: To modify an existing object (other than enabling a crossRef object), the client-supplied ENTINF structure includes ENTINF\_REMOTE\_MODIFY in the **ulFlags** field and specifies the modified attributes and their values. The client has to have the same rights as those needed to perform the modification via LDAP. The DC enforces the same schema and other constraints on the modification as if performed via LDAP. Performing the modification by using IDL\_DRSAddEntry rather than LDAP allows changes to multiple objects to be made in a single transaction.[<8>](#Appendix_A_8" \o "Product behavior note 8)

1. ULONG
2. IDL\_DRSAddEntry(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_ADDENTRYREQ \*pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DRS\_MSG\_ADDENTRYREPLY \*pmsgOut)
10. ext: DRS\_EXTENSIONS\_INT
11. pEntInfList: ADDRESS OF ENTINFLIST
12. pClientCreds: ADDRESS OF DRS\_SecBufferDesc
13. objCls : ATTRTYP
14. ncNameV: DSName
15. infoList: ADDENTRY\_REPLY\_INFO
16. cObjects: ULONG
17. res: boolean
18. prefixTable: PrefixTable
19. ValidateDRSInput(hDrs, 17)
20. /\* Only attributes and classes in the base schema can be specified.\*/
21. prefixTable := NewPrefixTable()
22. /\* Set the default response version \*/
23. pdwOutVersion := 2
24. if dwInVersion = 1 then /\* obsolete \*/
25. pmsgOut^.V1.Guid := 0
26. pmsgOut^.V1.Sid := 0
27. pmsgOut^.V1.errCode := 0
28. pmsgOut^.V1.dsid := 0
29. pmsgOut^.V1.extendedErr := 0
30. pmsgOut^.V1.extendedData := 0
31. pmsgOut^.V1.problem := 0
32. else if dwInVersion = 2 then
33. pmsgOut^.V2.pErrorObject:= null
34. pmsgOut^.V2.errCode := 0
35. pmsgOut^.V2.dsid := 0
36. pmsgOut^.V2.extendedEr := 0
37. pmsgOut^.V2.extendedData := 0
38. pmsgOut^.V2.problem := 0
39. pmsgOut^.V2.cObjectsAdded := 0
40. pmsgOut^.V2.infoList := null
41. else if dwInVersion = 3 then
42. pmsgOut^.V3.pdsErrObject := null
43. pmsgOut^.V3.dwErrVer := 0
44. pmsgOut^.V3. pErrData := null
45. pmsgOut^.V3.ULONG cObjectsAdded := 0
46. pmsgOut^.V3.infoList := null
47. endif
48. /\* Validate parameters. \*/
49. if not (dwInVersion in {2,3}) then
50. SetErrorData(SV\_PROBLEM\_UNAVAILABLE, 0, ERROR\_DS\_UNAVAILABLE,
51. pmsgOut, 2)
52. return 0
53. endif
54. /\* If the client supports the version 3 response, use version 3. \*/
55. ext := ClientExtensions(hDrs)
56. if DRS\_EXT\_ADDENTRYREPLY\_V3 in ext.dwFlags then
57. pdwOutVersion^ := 3
58. else
59. pdwOutVersion^ := 2
60. endif
61. cObjects := 0
62. if dwInVersion = 2 then
63. pEntInfList := pmsgIn^.V2.EntInfList
64. pClientCreds := null
65. else
66. pEntInfList := pmsgIn^.V3.EntInfList
67. pClientCreds := pmsgIn^.V3.pClientCreds
68. endif
69. /\* If explicit credentials are given, use them for access checks. \*/
70. if pClientCreds ≠ null then
71. err := UseCredsForAccessCheck(pClientCreds^)
72. if err ≠ 0 then
73. return err
74. endif
75. endif
76. /\* Walk through each item in the EntInfList and perform the requested
77. \* operation. \*/
78. e := pEntInfList
79. while e ≠ null
80. if ENTINF\_REMOTE\_MODIFY in e^.ulFlags then
81. if DSAObj()!msDS-Behavior-Version ≥ DS\_BEHAVIOR\_WIN2008 then
82. res := PerformModifyEntInf(
83. hDrs, e^.Entinf, ADR(infoList[cObjects]))
84. if not res then
85. return 0
86. endif
87. else
88. /\* Not supported (Win2k3 or older DC). \*/
89. SetErrorData(SV\_PROBLEM\_UNAVAILABLE,
90. 0,
91. ERROR\_DS\_UNAVAILABLE,
92. pmsgOut,
93. pdwOutVersion^)
94. return 0
95. endif
96. else
97. objCls := ENTINF\_GetValue(e^.Entinf, objectClass, prefixTable)
98. if objCls = crossRef then
99. /\* Create or enable a crossRef object. \*/
100. res := CreateCrossRef(hDrs, e^.Entinf, psmgOut, pdwOutVersion^,
101. ADR(infoList[cObjects]))
102. if not res then
103. return 0
104. endif
105. else if objCls = nTDSDSA then
106. /\* Create an nTDSDSA object. \*/
107. res := CreateNtdsDsa(hDrs, e^.Entinf, pEntInfList, pmsgOut,
108. pdwOutVersion^, ADR(infoList[cObjects]))
109. if not res then
110. return 0
111. endif
112. else
113. /\* Not supported. \*/
114. SetErrorData(SV\_PROBLEM\_BUSY, 0, ERROR\_DS\_DRA\_INVALID\_PARAMETER,
115. pmsgOut, pdwOutVersion^)
116. return 0
117. endif
118. endif
119. e := e^.pNextEntInf
120. cObjects := cObjects + 1
121. endwhile
122. if pdwOutVersion^ = 2 then
123. pmsgOut^.V2.cObjectsAdded := cObjects
124. pmsgOut^.V2.infoList := infoList
125. else
126. pmsgOut^.V3.cObjectsAdded := cObjects
127. pmsgOut^.V3.infoList := infoList
128. endif
129. return 0

### IDL\_DRSAddSidHistory (Opnum 20)

The IDL\_DRSAddSidHistory method adds one or more [**SIDs**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) to the sIDHistory [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of a given [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

1. ULONG IDL\_DRSAddSidHistory(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_ADDSIDREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_ADDSIDREPLY\* pmsgOut
9. );

**hDrs:** [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** Version of the request message. Must be set to 1, because no other version is supported.

**pmsgIn:** Pointer to the request message.

**pdwOutVersion:** Pointer to the version of the response message. The value must be 1, because no other version is supported.

**pmsgOut:** Pointer to the response message.

**Return Values:** 0 or one of the following [**Windows error codes**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b): ERROR\_DS\_MUST\_RUN\_ON\_DST\_DC or ERROR\_INVALID\_PARAMETER.

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_ADDSIDREQ

The DRS\_MSG\_ADDSIDREQ union defines the request messages that are sent to the [IDL\_DRSAddSidHistory](#Section_376230a5d8064ae5970af6243ee193c8) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_ADDSIDREQ\_V1 V1;
6. } DRS\_MSG\_ADDSIDREQ;

**V1:**  Version 1 request.

##### DRS\_MSG\_ADDSIDREQ\_V1

The DRS\_MSG\_ADDSIDREQ\_V1 structure defines the request message sent to the [IDL\_DRSAddSidHistory](#Section_376230a5d8064ae5970af6243ee193c8) method.

1. typedef struct {
2. DWORD Flags;
3. [string] WCHAR\* SrcDomain;
4. [string] WCHAR\* SrcPrincipal;
5. [string, ptr] WCHAR\* SrcDomainController;
6. [range(0,256)] DWORD SrcCredsUserLength;
7. [size\_is(SrcCredsUserLength)] WCHAR\* SrcCredsUser;
8. [range(0,256)] DWORD SrcCredsDomainLength;
9. [size\_is(SrcCredsDomainLength)]
10. WCHAR\* SrcCredsDomain;
11. [range(0,256)] DWORD SrcCredsPasswordLength;
12. [size\_is(SrcCredsPasswordLength)]
13. WCHAR\* SrcCredsPassword;
14. [string] WCHAR\* DstDomain;
15. [string] WCHAR\* DstPrincipal;
16. } DRS\_MSG\_ADDSIDREQ\_V1;

**Flags:**  A set of zero or more [DRS\_ADDSID\_FLAGS](#Section_76d50efed16542eeb8e45face33fe081) bit flags.

**SrcDomain:**  Name of the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) to query for the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of **SrcPrincipal**. The domain name can be an [**FQDN (1)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) or a NetBIOS name.

**SrcPrincipal:**  Name of a [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) (user, computer, or [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac)) in the source domain. This is the source security principal, whose SIDs will be added to the destination security principal. If **Flags** contains DS\_ADDSID\_FLAG\_PRIVATE\_CHK\_SECURE, this parameter is not used and is not validated. Otherwise, if **Flags** does not contain DS\_ADDSID\_FLAG\_PRIVATE\_DEL\_SRC\_OBJ, this name is a domain-relative Security Accounts Manager (SAM) name. Otherwise, it is a [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b).

**SrcDomainController:**  Name of the [**primary domain controller (PDC)**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) (or [**PDC role owner**](#gt_e42e52b3-e44f-4284-9c1b-e161f81ea516)) in the source domain. The [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) name can be an [**Internet host name**](#gt_4d5d5403-372f-4f9f-8d7a-65c310c807d9) or a NetBIOS name. This parameter is only used when **Flags** contains neither DS\_ADDSID\_FLAG\_PRIVATE\_CHK\_SECURE nor DS\_ADDSID\_FLAG\_PRIVATE\_DEL\_SRC\_OBJ. If **Flags** contains DS\_ADDSID\_FLAG\_PRIVATE\_DEL\_SRC\_OBJ, this parameter is not used, but it is validated.

**SrcCredsUserLength:**  Count of characters in the **SrcCredsUser** array.

**SrcCredsUser:**  User name for the credentials to be used in the source domain.

**SrcCredsDomainLength:**  Count of characters in the **SrcCredsDomain** array.

**SrcCredsDomain:**  [**Domain name**](#gt_45a1c9f1-0263-49a8-97c7-7aca1a99308c) for the credentials to be used in the source domain. The domain name can be an FQDN (1) or a [**NetBIOS domain name**](#gt_f7f8efcc-c6d5-40f0-9605-c9d99c5a0b92).

**SrcCredsPasswordLength:**  Count of characters in the **SrcCredsPassword** array.

**SrcCredsPassword:**  Password for the credentials to be used in the source domain.

**DstDomain:**  Name of the destination domain in which **DstPrincipal** resides. The domain name can be an FQDN (1) or a NetBIOS name.

**DstPrincipal:**  Name of a security principal (user, computer, or group) in the destination domain. This is the destination [**principal**](#gt_8492780e-99e2-47ba-8553-aedb8de9f9c0), to which the source principal's SIDs will be added. If **Flags** contains DS\_ADDSID\_FLAG\_PRIVATE\_CHK\_SECURE, this parameter is not used and is not validated. Otherwise, if **Flags** does not contain DS\_ADDSID\_FLAG\_PRIVATE\_DEL\_SRC\_OBJ, this name is a domain-relative SAM name. Otherwise, it is a DN.

##### DRS\_MSG\_ADDSIDREPLY

The DRS\_MSG\_ADDSIDREPLY union defines the response messages received from the [IDL\_DRSAddSidHistory](#Section_376230a5d8064ae5970af6243ee193c8) method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_ADDSIDREPLY\_V1 V1;
6. } DRS\_MSG\_ADDSIDREPLY;

**V1:**  Version 1 of the reply packet structure.

##### DRS\_MSG\_ADDSIDREPLY\_V1

The DRS\_MSG\_ADDSIDREPLY\_V1 structure defines the response message received from the [IDL\_DRSAddSidHistory](#Section_376230a5d8064ae5970af6243ee193c8) method.

1. typedef struct {
2. DWORD dwWin32Error;
3. } DRS\_MSG\_ADDSIDREPLY\_V1;

**dwWin32Error:**  Zero if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

##### DRS\_ADDSID\_FLAGS

The DRS\_ADDSID\_FLAGS type consists of bit flags that indicate how the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) is to be added to the [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409).

The valid bit flags are shown in the following diagram. The flags are represented in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | D E L | C S | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**CS (DS\_ADDSID\_FLAG\_PRIVATE\_CHK\_SECURE, 0x40000000)**: If set, the server verifies whether the channel is secure and returns the result of the verification in the response.

**DEL (DS\_ADDSID\_FLAG\_PRIVATE\_DEL\_SRC\_OBJ, 0x80000000)**: If set, the server appends the objectSid and sIDHistory [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of SrcPrincipal to the sIDHistory attribute of DstPrincipal, and deletes SrcPrincipal from the source [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

This type is declared as follows:

1. typedef DWORD DRS\_ADDSID\_FLAGS;

#### Method-Specific Abstract Types and Procedures

##### ConnectionCtx

The ConnectionCtx [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) represents a connection to a specific server with a given set of credentials. It does not imply any particular protocol or transport. It provides a means for pseudocode to compactly represent the notion of the target server and corresponding credentials for an operation.

Procedures that take a ConnectionCtx as an input perform their operations against the server represented by the ConnectionCtx, using the credentials associated with the ConnectionCtx.

##### ConnectToDC

1. procedure ConnectToDC(dcname: unicodestring): ConnectionCtx

Creates a [ConnectionCtx](#Section_1f7499b2debd48bc8f5fdd4e12576216) for the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) named by *dcname*, associating the credentials of the client's [**security context**](#gt_88d49f20-6c95-4b64-a52c-c3eca2fe5709), which MUST be retrieved using the method described in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.3.4.3, with the ConnectionCtx. *dcname* can be the [**Internet host name**](#gt_4d5d5403-372f-4f9f-8d7a-65c310c807d9) or the NetBIOS name of the DC. If the ConnectionCtx cannot be created, the procedure returns null.

##### ConnectToDCWithCreds

1. procedure ConnectToDCWithCreds(
2. dcname: unicodestring,
3. username: unicodestring,
4. pwd: unicodestring,
5. domain: unicodestring): ConnectionCtx

Creates a [ConnectionCtx](#Section_1f7499b2debd48bc8f5fdd4e12576216) for the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) named by *dcname*, associating the credentials of user *username*, password *pwd*, and user-domain [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) with the ConnectionCtx. *dcname* can be the [**Internet host name**](#gt_4d5d5403-372f-4f9f-8d7a-65c310c807d9) or the NetBIOS name of the DC. If the ConnectionCtx cannot be created, it returns null.

##### GenerateFailureAudit

1. procedure GenerateFailureAudit()

Generates a failure audit event on the server on which it is called if auditing is enabled. The generated audit event indicates that an operation failed. This procedure does nothing if auditing is not enabled. The content of the audit event is an implementation-specific behavior.

##### GenerateSuccessAudit

1. procedure GenerateSuccessAudit()

Generates a success audit event on the server on which it is called if auditing is enabled. The generated audit event indicates that an operation succeeded. This procedure does nothing if auditing is not enabled. The content of the audit event is an implementation-specific behavior.

##### GenerateSuccessAuditRemotely

1. procedure GenerateSuccessAuditRemotely(ctx: ConnectionCtx): boolean

If auditing is enabled on the server associated with *ctx*, the GenerateSuccessAuditRemotely procedure generates a success audit event on that server and returns true. The generated audit event indicates that an operation succeeded. Returns false if auditing is not enabled on that server. The content of the audit event is an implementation-specific behavior.[<9>](#Appendix_A_9" \o "Product behavior note 9)

##### GetKeyLength

1. procedure GetKeyLength(hDrs: DRS\_HANDLE): integer

Returns the key length, in bits, of the encryption used on the *hDrs* connection. Returns 0 if no encryption is in use on the connection.

##### FindGC

1. procedure FindGC(): unicodestring

Returns the [**Internet host name**](#gt_4d5d5403-372f-4f9f-8d7a-65c310c807d9) of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that is a [**GC server**](#gt_a5a99ce4-e206-42dc-8874-e103934c5b0d) in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) (see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.3.6), or null if such a DC cannot be found.

##### GetPDC

1. procedure GetPDC(domainName: unicodestring): unicodestring

Returns the [**Internet host name**](#gt_4d5d5403-372f-4f9f-8d7a-65c310c807d9) of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that holds the [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f) for the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) whose name is *domainName* (see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.5.4), or null if such a DC cannot be found. *domainName* can be either the [**FQDN (1)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) or the NetBIOS name of the domain.

##### HasAdminRights

1. procedure HasAdminRights(ctx: ConnectionCtx) : boolean

Returns true if the credentials associated with *ctx* have administrative rights on the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) associated with *ctx*. Possessing administrative rights is defined as having the ability to write to (that is, change the membership of) the Domain Admins [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) in the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) that is the default [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) on the DC associated with *ctx*.

##### IsAuditingEnabled

1. procedure IsAuditingEnabled (): boolean

Returns true if auditing on the server on which it is called is enabled, and returns false otherwise.

##### IsLocalRpcCall

1. procedure IsLocalRpcCall(hDrs: DRS\_HANDLE): boolean

Returns true if the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) call that is being processed on *hDrs* originated from the same computer as the computer that is processing the call.

##### IsNT4SP4OrBetter

1. procedure IsNT4SP4OrBetter(ctx: ConnectionCtx): boolean

If the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) named in *ctx* is running Windows NT 4.0 and is not running at least Windows NT 4.0 operating system Service Pack 4 (SP4), this procedure returns false. Otherwise, it returns true.[<10>](#Appendix_A_10" \o "Product behavior note 10)

##### IsAuditingGroupPresent

1. procedure IsAuditingGroupPresent(dcname: unicodestring, nETBIOSName: unicodestring): DWORD

Returns ERROR\_NO\_SUCH\_ALIAS if the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) represented by the *dcname* does not have a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) local [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) whose sAMAccountName is the value of the *nETBIOSName* parameter appended with three dollar signs $$$. Otherwise, it returns ERROR\_SUCCESS. This group is not present by default and must be created by the administrator of the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) service.

##### IsWellKnownDomainRelativeSid

1. procedure IsWellKnownDomainRelativeSid(sid: SID): boolean

Returns true if *sid* consists of the [**domain SID**](#gt_c1d6ba4d-2302-43a5-acd2-02bfe19d0ade) of the server's default [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) and of a [**RID**](#gt_df3d0b61-56cd-4dac-9402-982f1fedc41c) (as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2) whose value is less than 1000, and returns false otherwise.

##### LastRID

1. procedure LastRID(sid: SID): Rid

Extracts and returns the [**RID**](#gt_df3d0b61-56cd-4dac-9402-982f1fedc41c) from the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) *sid*. See [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2.

##### RemoteQuery

1. procedure RemoteQuery(
2. ctx: ConnectionCtx,
3. query: unicodestring): select-return-value

Performs the select statement represented by the string *query* against the server associated with *ctx*, using the credentials associated with *ctx*. Returns the results of the select operation. The return value of this function is the same type as the return value of the select statement performed.

#### Server Behavior of the IDL\_DRSAddSidHistory Method

*Informative summary of behavior*: The [IDL\_DRSAddSidHistory](#Section_376230a5d8064ae5970af6243ee193c8) method adds the [**SIDs**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) associated with one [**principal**](#gt_8492780e-99e2-47ba-8553-aedb8de9f9c0) (the source principal) to the sIDHistory [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of another principal (the destination principal). The source principal's objectSid and any SIDs in the source principal's sIDHistory are added to the destination principal's sIDHistory. This method is called on a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) whose [**default NC**](#gt_adc2c434-9679-419f-8c8b-b8c234921ad3) contains the destination principal. If necessary, the destination DC will contact a DC whose default NC contains the source principal as part of executing this method.

This method has three different variants on this behavior, and the caller indicates which variant is desired by specifying a combination of flags in pmsgIn^.V1.flags.

* If the DS\_ADDSID\_FLAG\_PRIVATE\_CHK\_SECURE flag is specified, the first variant is selected. In this variant, the method verifies only that the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) call is secure. It does not perform any further processing or manipulate the sIDHistory attribute of any [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca), regardless of other flags that might be present.
* If DS\_ADDSID\_FLAG\_PRIVATE\_CHK\_SECURE is not specified but DS\_ADDSID\_FLAG\_PRIVATE\_DEL\_SRC\_OBJ is specified, the second variant is selected. In this variant, the source and destination principals are in the same [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). The values of the objectSid and sIDHistory attributes of the source principal are added to the destination principal's sIDHistory attribute, and then the source principal is deleted. See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.5.5 for more information about the delete operation. Loosely speaking, the destination principal adopts the source principal as an "alias" and the source principal disappears.
* The third variant is selected by specifying neither DS\_ADDSID\_FLAG\_PRIVATE\_CHK\_SECURE nor DS\_ADDSID\_FLAG\_PRIVATE\_DEL\_SRC\_OBJ. In this variant, the source and destination principals are in different [**forests**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). The values of the source principal's objectSid and sIDHistory attributes are copied into the destination principal's sIDHistory attribute, as in the second variant, but without deleting the source principal. Loosely speaking, the destination principal adopts the source principal as an "alias" while coexisting with the source principal.

The preceding are the only variants supported by the IDL\_DRSAddSidHistory method. In particular, the case of source and destination principals in different domains within the same forest is not supported.

1. ULONG
2. IDL\_DRSAddSidHistory(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_ADDSIDREQ \*pmsgIn,
6. [out, ref] DWORD \*pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)] DRS\_MSG\_ADDSIDREPLY \*pmsgOut)
8. flags: DRS\_ADDSID\_FLAGS
9. srcPrinc: DSName
10. dstPrinc: DSName
11. srcPrincInDst: DSName
12. srcNc: DSName
13. dstNc: DSName
14. crSrc: DSName
15. crDst: DSName
16. partCtr: DSName
17. srcDomainController: unicodestring
18. srcCtx: ConnectionCtx
19. srcPrincSid: SID
20. srcPrincSidHistory: set of SID
21. rt: ULONG
22. ValidateDRSInput(hDrs, 20)
23. pdwOutVersion^ := 1
24. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_INTERNAL\_FAILURE
25. flags := pmsgIn^.V1.flags
26. if DS\_ADDSID\_FLAG\_PRIVATE\_CHK\_SECURE in flags then
27. /\* First mode of operation: verify connection security.
28. \* If connecting from off-machine, connection must have 128-bit
29. \* encryption or better. \*/
30. if (not IsLocalRpcCall(hDrs)) and
31. (GetKeyLength(hDrs) < 128) then
32. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_MUST\_RUN\_ON\_DST\_DC
33. return ERROR\_DS\_MUST\_RUN\_ON\_DST\_DC
34. else
35. return 0
36. endif
37. endif
38. /\* Currently, only version 1 is supported. The RPC IDL definitions
39. \* for the interface do not allow passing in a version other than 1. \*/
40. if dwInVersion ≠ 1 then
41. return ERROR\_INVALID\_PARAMETER
42. endif
43. if DS\_ADDSID\_FLAG\_PRIVATE\_DEL\_SRC\_OBJ in flags then
44. /\* Second mode of operation: add objectSid/sidHistory from source
45. \* principal to destination principal, then delete source
46. \* principal. \*/
47. /\* Basic parameter validation \*/
48. if (pmsgIn^.V1.SrcDomain ≠ null) or
49. (pmsgIn^.V1.DstDomain ≠ null) or
50. (pmsgIn^.V1.SrcCredsUserLength ≠ 0) or
51. (pmsgIn^.V1.SrcCredsDomainLength ≠ 0) or
52. (pmsgIn^.V1.SrcCredsPasswordLength ≠ 0) or
53. (pmsgIn^.V1.SrcDomainController = "") or
54. (pmsgIn^.V1.SrcPrincipal = null) or
55. (pmsgIn^.V1.SrcPrincipal = "") or
56. (pmsgIn^.V1.DstPrincipal = null) or
57. (pmsgIn^.V1.DstPrincipal = "") then
58. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_INTERNAL\_FAILURE
59. return ERROR\_INVALID\_PARAMETER
60. endif
61. /\* In this case, pmsgIn^.V1.SrcPrincipal and .DstPrincipal are
62. \* DNs. \*/
63. srcPrinc := GetDSNameFromDN(pmsgIn^.V1.SrcPrincipal)
64. dstPrinc := GetDSNameFromDN(pmsgIn^.V1.DstPrincipal)
65. srcNc := GetObjectNC(srcPrinc)
66. dstNc := GetObjectNC(dstPrinc)
67. /\* Source and destination principals must be in same domain. \*/
68. if srcNc = null or dstNc = null or srcNc ≠ dstNc then
69. pmsgOut^.V1.dwWin32Error := ERROR\_INVALID\_PARAMETER
70. return 0
71. endif
72. /\* Destination NC must be this server's default domain NC. \*/
73. if dstNc ≠ DefaultNC() then
74. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_MASTERDSA\_REQUIRED
75. return 0
76. endif
77. /\* Verify that this server has auditing enabled \*/
78. if not IsAuditingEnabled () then
79. pmsgOut^.V1.dwWin32Error :=
80. ERROR\_DS\_DESTINATION\_AUDITING\_NOT\_ENABLED
81. return 0
82. endif
83. /\* Must have the control access right. \*/
84. if not AccessCheckCAR(dstNc, Migrate-SID-History) then
85. GenerateFailureAudit()
86. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_INSUFF\_ACCESS\_RIGHTS
87. return 0
88. endif
89. /\* Destination domain must be in native mode. \*/
90. partCtr := DescendantObject(ConfigNC(), "CN=Partitions,")
91. if partCtr ≠ null
92. crDst := select one dd from subtree partCtr where
93. (crossRef in dd!objectClass and
94. dd!nCName = dstNc)
95. endif
96. if partCtr = null or crDst = null then
97. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_INTERNAL\_FAILURE
98. return 0
99. else
100. if crDst!nTMixedDomain = 1 then
101. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_DST\_DOMAIN\_NOT\_NATIVE
102. return 0
103. endif
104. endif
105. /\* Validation of object state. \*/
106. if (not ObjExists(srcPrinc)) or
107. (not (user in srcPrinc!objectClass or
108. group in srcPrinc!objectClass)) or
109. (not ObjExists(dstPrinc)) or
110. (not (user in dstPrinc!objectClass or
111. group in dstPrinc!objectClass)) or
112. (srcPrinc = dstPrinc) or
113. (IsWellKnownDomainRelativeSid(srcPrinc!objectSid)) or
114. (IsWellKnownDomainRelativeSid(dstPrinc!objectSid)) then
115. pmsgOut^.V1.dwWin32Error := ERROR\_INVALID\_PARAMETER
116. return 0
117. endif
118. /\* Check that this machine has rights to delete the source principal. \*/
119. if (not AccessCheckObject(srcPrinc, RIGHT\_DELETE)) and
120. (not AccessCheckObject(srcPrinc.parent, RIGHT\_DS\_DELETE\_CHILD))
121. then
122. pmsgOut^.V1.dwWin32Error := ERROR\_ACCESS\_DENIED
123. return 0
124. endif
125. /\* Save the source principal's SID and SID history and then delete the principal \*/
126. srcPrincSid := srcPrinc!objectSid
127. srcPrincSidHistory := srcPrinc!sIDHistory
128. rt = RemoveObj(srcPrinc,false)
129. if(rt ≠ 0) then
130. pmsgOut^.V1.dwWin32Error := rt
131. return 0
132. endif
133. /\* Add source principal's objectSid and sidHistory to
134. \* destination principal's sidHistory. \*/
135. dstPrinc!sidHistory := dstPrinc!sidHistory + {srcPrincSid}
136. dstPrinc!sidHistory := dstPrinc!sidHistory + srcPrincSidHistory
137. GenerateSuccessAudit()
138. return 0
139. endif
140. /\* Third mode of operation: add objectSid/sIDHistory from source
141. \* principal to destination principal. Source principal is
142. \* untouched. \*/
143. /\* Basic parameter validation. \*/
144. if (pmsgIn^.V1.SrcDomain = null) or
145. (pmsgIn^.V1.SrcDomain = "") or
146. (pmsgIn^.V1.DstDomain = null) or
147. (pmsgIn^.V1.DstDomain = "") or
148. (pmsgIn^.V1.SrcCredsUserLength > 0 and
149. pmsgIn^.V1.SrcCredsUser = null) or
150. (pmsgIn^.V1.SrcCredsDomainLength > 0 and
151. pmsgIn^.V1.SrcCredsDomain = null) or
152. (pmsgIn^.V1.SrcCredsPasswordLength > 0 and
153. pmsgIn^.V1.SrcCredsPassword = null) or
154. (pmsgIn^.V1.SrcDomainController = "") or
155. (pmsgIn^.V1.SrcPrincipal = null) or
156. (pmsgIn^.V1.SrcPrincipal = "") or
157. (pmsgIn^.V1.DstPrincipal = null) or
158. (pmsgIn^.V1.DstPrincipal = "") then
159. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_INTERNAL\_FAILURE
160. return ERROR\_INVALID\_PARAMETER
161. endif
162. /\* Confirm destination domain is in forest of server. \*/
163. crDst := select one dd from subtree ConfigNC() where
164. (crossRef in dd!objectClass and
165. (dd!dnsRoot = pmsgIn^.V1.DstDomain or
166. dd!nETBIOSName = pmsgIn^.V1.DstDomain))
167. if crDst = null then
168. pmsgOut^.V1.dwWin32Error :=
169. ERROR\_DS\_DESTINATION\_DOMAIN\_NOT\_IN\_FOREST
170. return 0
171. endif
172. /\* Confirm source domain is not in forest of server. \*/
173. crSrc := select one ss from subtree ConfigNC() where
174. (crossRef in ss!objectClass and
175. (ss!dnsRoot = pmsgIn^.V1.SrcDomain or
176. ss!nETBIOSName = pmsgIn^.V1.SrcDomain)
177. and FLAG\_CR\_NTDS\_NC in ss!systemFlags
178. and FLAG\_CR\_NTDS\_DOMAIN in ss!systemFlags)
179. if crSrc ≠ null then
180. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_SOURCE\_DOMAIN\_IN\_FOREST
181. return 0
182. endif
183. /\* Destination NC must be this server's default domain NC. \*/
184. if crDst!nCName ≠ DefaultNC() then
185. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_MASTERDSA\_REQUIRED
186. return 0
187. endif
188. /\* Destination domain must be in native mode. \*/
189. if crDst!nTMixedDomain = 1 then
190. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_DST\_DOMAIN\_NOT\_NATIVE
191. return 0
192. endif
193. dstNC := crDst!nCName
194. /\* Verify this server has auditing enabled for destination domain. \*/
195. if not IsAuditingEnabled () then
196. pmsgOut^.V1.dwWin32Error :=
197. ERROR\_DS\_DESTINATION\_AUDITING\_NOT\_ENABLED
198. return 0
199. endif
200. /\* Must have the control access right. \*/
201. if not AccessCheckCAR(dstNc, Migrate-SID-History) then
202. GenerateFailureAudit()
203. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_INSUFF\_ACCESS\_RIGHTS
204. return 0
205. endif
206. /\* Retrieve destination principal.
207. \* In this case, pmsgIn^.V1.DstPrincipal is a SAM name. \*/
208. dstPrinc := select one o from subtree DefaultNC() where
209. (o!sAMAccountName = pmsgIn^.V1.DstPrincipal)
210. if dstPrinc = null then
211. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_OBJ\_NOT\_FOUND
212. return 0
213. endif
214. /\* Locate a source DC if one wasn't supplied. Source DC must be
215. \* the PDC FSMO role owner. \*/
216. srcDomainController := pMsgin^.V1.SrcDomainController
217. if srcDomainController = null then
218. srcDomainController := GetPDC(pmsgIn^.V1.SrcDomain)
219. else
220. if srcDomainController ≠ GetPDC(pmsgIn^.V1.SrcDomain) then
221. pmsgOut^.V1.dwWin32Error := ERROR\_INVALID\_DOMAIN\_ROLE
222. return 0
223. endif
224. endif
225. if srcDomainController = null then
226. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_CANT\_FIND\_DC\_FOR\_SRC\_DOMAIN
227. return 0
228. endif
229. /\* Connect to source DC, using supplied credentials if applicable. \*/
230. if (pmsgIn^.V1.SrcCredsUserLength = 0) and
231. (pmsgIn^.V1.SrcCredsPasswordLength = 0) and
232. (pmsgIn^.V1.SrcCredsDomainLength = 0) then
233. srcCtx := ConnectToDC(srcDomainController)
234. else
235. srcCtx := ConnectToDCWithCreds(srcDomainController,
236. pmsgIn^.V1.SrcCredsUser, pmsgIn^.V1.SrcCredsPassword,
237. pmsgIn^.V1.SrcCredsDomain)
238. endif
239. if (srcCtx = null) then
240. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_CANT\_FIND\_DC\_FOR\_SRC\_DOMAIN
241. return 0
242. endif
243. /\* Confirm client has administrative rights on source DC. \*/
244. if not HasAdminRights(srcCtx) then
245. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_INSUFF\_ACCESS\_RIGHTS
246. return 0
247. endif
248. /\* Retrieve source principal from source DC using the remote connection.
249. \* In this case, pmsgIn^.V1.SrcPrincipal is a SAM name.
250. \* Example: If pmsgIn^.V1.SrcPrincipal value is username1 then
251. \* following query is executed in the source DC:
252. \* select one o from subtree dc.defaultNC where (o!sAMAccountName = "username1")
253. \*/
254. srcPrinc := RemoteQuery(srcCtx,
255. "select one o from subtree dc.defaultNC where (o!sAMAccountName = "
256. + '"' + pmsgIn^.V1.SrcPrincipal + '"' + ")"
257. )
258. if srcPrinc = null then
259. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_OBJ\_NOT\_FOUND
260. return 0
261. endif
262. /\* Source principal must be user (which includes computer) or
263. \* group.\*/
264. if not (group in srcPrinc!objectClass or
265. user in srcPrinc!objectClass) then
266. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_SRC\_OBJ\_NOT\_GROUP\_OR\_USER
267. return 0
268. endif
269. srcPrincSid := srcPrinc!objectSid
270. srcPrincSidHistory := srcPrinc!sIDHistory
271. /\* Verify that no principal other than the destination
272. \* principal exists in the destination forest that contains
273. \* a SID that matches the source principal. \*/
274. if IsGC() or IsAdlds() then
275. srcPrincInDst := select one o from subtree DefaultNC() where
276. (o ≠ dstPrinc) and
277. ((o!objectSid = srcPrincSid) or
278. (o!objectSid in srcPrincSidHistory) or
279. (srcPrincSid in o!sIDHistory)) or
280. ((srcPrincSidHistory ∩ o!sIDHistory) ≠ {}))
281. if srcPrincInDst ≠ null then
282. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_SRC\_SID\_EXISTS\_IN\_FOREST
283. return 0
284. endif
285. else
286. /\* The current DC is not a GC server.
287. \* We need to locate a GC server and perform an IDL\_DRSCrackNames query against it in order
288. for the SID search to be Forest scoped \*/
289. gcDomainController : unicodestring
290. hDrsGc: DRS\_HANDLE
291. crackMsgIn: DRS\_MSG\_CRACKREQ\_V1
292. crackOut: DS\_NAME\_RESULTW
293. gcDomainController := FindGC()
294. if gcDomainController = null then
295. return STATUS\_DS\_GC\_NOT\_AVAILABLE
296. endif
297. /\* Bind to GC \*/
298. hDrsGc := BindToDSA(gcDomainController)
299. if hDrsGc = null then
300. pmsgOut^.V1.dwWin32Error := STATUS\_DS\_GC\_NOT\_AVAILABLE
301. return 0
302. endif
303. crackMsgIn.dwFlags := DS\_NAME\_FLAG\_GCVERIFY
304. crackMsgIn.formatOffered := DS\_STRING\_SID\_NAME
305. crackMsgIn.formatDesired := DS\_UNIQUE\_ID\_NAME
306. crackMsgIn.cNames := 1
307. crackMsgIn.rpNames[0] := srcPrincSid
308. crackNamesErr := IDL\_DRSCrackNames(
309. hDrsGc,
310. dwInVersion,
311. crackMsgIn,
312. pdwOutVersion,
313. ADR(crackOut))
314. if crackNamesErr ≠ 0 then
315. if crackNamesErr = DS\_NAME\_ERROR\_NOT\_UNIQUE then
316. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_SRC\_SID\_EXISTS\_IN\_FOREST
317. return 0
318. elseif crackNamesErr ≠ DS\_NAME\_ERROR\_NOT\_FOUND and
319. crackNamesErr ≠ DS\_NAME\_ERROR\_DOMAIN\_ONLY then
320. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_INTERNAL\_FAILURE
321. return 0
322. endif
323. if crackOut.rItems ≠ null and
324. crackOut.rItems[0].pName ≠ dstPrinc!objectGUID then
325. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_SRC\_SID\_EXISTS\_IN\_FOREST
326. return 0
327. endif
328. UnbindFromDSA(hDrsGc)
329. endif
330. /\* Confirm source domain has auditing enabled and generate an audit
331. \* event on it. \*/
332. if not GenerateSuccessAuditRemotely(srcCtx)
333. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_SOURCE\_AUDITING\_NOT\_ENABLED
334. return 0
335. endif
336. /\* Verify that if source domain is running Windows NT 4.0, it is
337. \* running at least Service Pack 4 of that operating system. \*/
338. if not IsNT4SP4OrBetter(srcCtx)
339. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_SRC\_DC\_MUST\_BE\_SP4\_OR\_GREATER
340. return 0
341. endif
342. /\* Verify that if source domain has a domain local group srcDomainNetBIOSName$$$
343. \*/
344. if IsAuditingGroupPresent(srcDomainController, pmsgIn^.V1.SrcDomain) = ERROR\_NO\_SUCH\_ALIAS
345. pmsgOut^.V1.dwWin32Error := ERROR\_NO\_SUCH\_ALIAS
346. return 0
347. endif
348. /\* Source and destination principals must both be computer, or both
349. \* be user, or both be group. The order is important: although
350. \* computer objects are user objects, the case is disallowed where
351. \* one principal is a computer and the other principal is a user
352. \* but not a computer. \*/
353. if ((computer in srcPrinc!objectClass and
354. not computer in dstPrinc!objectClass) or
355. (computer in dstPrinc!objectClass and
356. not computer in srcPrinc!objectClass)) or
357. ((user in srcPrinc!objectClass and
358. not user in dstPrinc!objectClass) or
359. (user in dstPrinc!objectClass and
360. not user in srcPrinc!objectClass)) or
361. ((group in srcPrinc!objectClass and
362. not group in dstPrinc!objectClass) or
363. (group in dstPrinc!objectClass and
364. not group in srcPrinc!objectClass)) then
365. pmsgOut^.V1.dwWin32Error :=
366. ERROR\_DS\_SRC\_AND\_DST\_OBJECT\_CLASS\_MISMATCH
367. return 0
368. endif
369. /\* Class-specific object state tests.
370. \* Note that computer is a subclass of user, so the following test
371. \* applies to both user and computer objects. \*/
372. if user in srcPrinc!objectClass then
373. if srcPrinc!userAccountControl ∩ {ADS\_UF\_NORMAL\_ACCOUNT,
374. ADS\_UF\_WORKSTATION\_TRUST\_ACCOUNT,
375. ADS\_UF\_SERVER\_TRUST\_ACCOUNT} ≠
376. dstPrinc!userAccountControl ∩ {ADS\_UF\_NORMAL\_ACCOUNT,
377. ADS\_UF\_WORKSTATION\_TRUST\_ACCOUNT,
378. ADS\_UF\_SERVER\_TRUST\_ACCOUNT} then
379. pmsgOut^.V1.dwWin32Error :=
380. ERROR\_DS\_SRC\_AND\_DST\_OBJECT\_CLASS\_MISMATCH
381. return 0
382. endif
383. if group in srcPrinc!objectClass and
384. srcPrinc!groupType ≠ dstPrinc!groupType then
385. pmsgOut^.V1.dwWin32Error :=
386. ERROR\_DS\_SRC\_AND\_DST\_OBJECT\_CLASS\_MISMATCH
387. return 0
388. endif
389. /\* Check if source principal is built-in principal. \*/
390. if IsBuiltinPrincipal(srcPrinc!objectSid) then
391. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_UNWILLING\_TO\_PERFORM
392. return 0
393. endif
394. /\* If source principal has well-known domain-relative SID
395. \* make sure final RIDs of source and destination principals
396. \* are the same. \*/
397. if IsWellKnownDomainRelativeSid(srcPrinc!objectSid) then
398. if LastRID(srcPrinc!objectSid) ≠ LastRID(dstPrinc!objectSid)
399. pmsgOut^.V1.dwWin32Error := ERROR\_DS\_UNWILLING\_TO\_PERFORM
400. return 0
401. endif
402. endif
403. /\* Add source principal's objectSid and sIDHistory to
404. \* destination principal's sidHistory. \*/
405. dstPrinc!sIDHistory := dstPrinc!sIDHistory + {srcPrincSid}
406. dstPrinc!sIDHistory := dstPrinc!sIDHistory + srcPrincSidHistory
407. GenerateSuccessAudit()
408. return 0

#### Examples of the IDL\_DRSAddSidHistory Method

##### Calling IDL\_DRSAddSidHistory with DS\_ADDSID\_FLAG\_PRIVATE\_CHK\_SECURE Flags

This flag is used when the caller wants to check whether an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) call to DC1 is secure.

###### Client Request

A client invokes the IDL\_DRSAddSidHistory method against DC1 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 1
* *pmsgIn* = DRS\_MSG\_ADDSIDREQ\_V1
  + Flags = 0x40000000

###### Server Response

The server returns a code of 0 and the following values:

* *pdwOutVersion* = 1
* *pmsgOut* = DRS\_MSG\_ADDSIDREPLY\_V1
  + dwWin32Error: 0

###### Final State

There are no changes in state.

##### Calling IDL\_DRSAddSidHistory with DS\_ADDSID\_FLAG\_PRIVATE\_DEL\_SRC\_OBJ Flags

In this example, the user "Kim Akers" has an account in [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) DC=contoso, DC=com with a Windows NT 4.0 account name "CONTOSO\kimakers". There is another account in the same domain for the user "Kim Akers" with the Windows NT 4.0 account name "CONTOSO\kimakers1". The domain administrator wants to add a [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of "CONTOSO\Kimakers1" account to the SIDHistory of "CONTOSO\kimakers" and delete "CONTOSO\Kimakers1".

###### Initial State

Querying the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) whose sAMAccountName is kimakers in the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) DC=CONTOSO, DC=COM on DC1:

* ldap\_search\_s("DC=contoso,DC=com", *wholeSubtree*, "(sAMAccountName=kimakers)", [*objectClass*, *distinguishedName*, *sAMAccountName*, *objectSid*, *sIDHistory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entry:
* >> Dn: CN=Kim Akers,CN=Users,DC=contoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> distinguishedName: CN=Kim Akers,CN=Users,DC=contoso,DC=com;
  + 1> sAMAccountName: KimAkers;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1144

Querying the user object whose sAMAccountName is kimakers1 in the domain NC DC=CONTOSO, DC=COM on DC1:

* ldap\_search\_s("DC=contoso,DC=com", *wholeSubtree*, "(sAMAccountName=kimakers1)", [*objectClass*, *distinguishedName*, *sAMAccountName*, *objectSid*, *sIDHistory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=Kim Akers1,CN=Users,DC=contoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> distinguishedName: CN=Kim Akers1,CN=Users,DC=contoso,DC=com;
  + 1> sAMAccountName: KimAkers1;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1129;

###### Client Request

A client invokes the **IDL\_DRSAddSidHistory** method against DC1 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 1
* *pmsgIn* = DRS\_MSG\_ADDSIDREQ\_V1
  + Flags = 0x80000000
  + SrcPrincipal = "CN=Kim Akers1,CN=Users,DC=contoso,DC=com"
  + DstPrincipal = "CN=Kim Akers,CN=Users,DC=contoso,DC=com"

###### Server Response

The server returns a code of 0 and the following values:

* *pdwOutVersion* = 1
* *pmsgOut* = DRS\_MSG\_ADDSIDREPLY\_V1
  + dwWin32Error: 0

###### Final State

The **sIDHistory** [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) whose [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) is "Kim Akers,CN=Users,DC=contoso,DC=com" contains one value:

* ldap\_search\_s("DC=contoso,DC=com", *wholeSubtree*, "(sAMAccountName=kimakers)", [*objectClass*, *distinguishedName*, *sAMAccountName*, *objectSid*, *sIDHistory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entry:
* >> Dn: CN=Kim Akers,CN=Users,DC=contoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> distinguishedName: CN=Kim Akers,CN=Users,DC=contoso,DC=com;
  + 1> sAMAccountName: KimAkers;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1144;
  + 1> sIDHistory: S-1-5-21-254470460-2440132622-709970653-1129;

The user object whose DN is "Kim Akers1,CN=Users,DC=contoso,DC=com" is deleted:

ldap\_search\_s("DC=contoso,DC=com", *wholeSubtree*, "(sAMAccountName=kimakers1)", [*objectClass*, *distinguishedName*, *sAMAccountName*, *objectSid*, *sIDHistory*])

* Result <0>: (null)
* Matched DNs:
* Getting 0 entries:

##### Calling IDL\_DRSAddSidHistory with 0 in Flags

The user "Kim Akers" has an account in [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) DC=contoso, DC=com with a Windows NT 4.0 account name "CONTOSO\kimakers". The user has another account in a separate [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) in domain DC=legacycontoso,DC=com with a Windows NT 4.0 account name "LEGACYCONTOSO\kimakers1". The domain administrator wants to add the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of "LEGACYCONTOSO\Kimakers1" account to the sIDHistory of "CONTOSO\kimakers". The administrator's account name in the LEGACYCONTOSO domain is LegacyContosoAdmin with password Passw0rd123. LEGACYCONTOSO is the NetBIOS name for the LEGACYCONTOSO.com domain.

###### Initial State

Querying the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) whose sAMAccountName is kimakers in the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) DC=CONTOSO, DC=COM on DC1:

* ldap\_search\_s("DC=contoso,DC=com", *wholeSubtree*, "(sAMAccountName=kimakers)", [*objectClass*, *distinguishedName*, *sAMAccountName*, *objectSid*, *sIDHistory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entry:
* Dn: CN=Kim Akers,CN=Users,DC=contoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> distinguishedName: CN=Kim Akers,CN=Users,DC=contoso,DC=com;
  + 1> sAMAccountName: KimAkers
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1144

Querying the user object whose sAMAccountName is kimakers1 in the domain NC DC=LEGACYCONTOSO, DC=COM on DC9:

* ldap\_search\_s("DC=legacycontoso,DC=com", wholeSubtree, "(sAMAccountName=kimakers1)", [*objectClass*, *distinguishedName*, *sAMAccountName*, *objectSid*, *sIDHistory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=Kim Akers1,CN=Users,DC=legacycontoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> distinguishedName: CN=Kim Akers1,CN=Users,DC=legacycontoso,DC=com;
  + 1> sAMAccountName: KimAkers1;
  + 1> objectSid: S-1-5-21-1137440724-3092688314-3181763971-1153;

###### Client Request

A client invokes the IDL\_DRSAddSidHistory method against DC1 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 1
* *pmsgIn* = DRS\_MSG\_ADDSIDREQ\_V1
  + Flags = 0
  + SrcDomain = "legacycontoso.com"
  + SrcPrincipal = "KimAkers1"
  + SrcCredsDomain = "legacycontoso"
  + SrcCredsDomainLength = 13
  + SrcCredsUser = "LegacyContosoAdmin"
  + SrcCredsUserLength = 18
  + SrcCredsPassword = "Passw0rd123"
  + SrcCredsPasswordLength = 11
  + DstDomain = "contoso"
  + DstPrincipal = "KimAkers"

###### Server Response

The server returns a code of 0 and the following values:

* *pdwOutVersion* = 1

*pmsgOut* = DRS\_MSG\_ADDSIDREPLY\_V1

* + dwWin32Error: 0

###### Final State

The sIDHistory [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) whose [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) is "Kim Akers,CN=Users,DC=contoso,DC=com" contains one value:

* ldap\_search\_s("DC=contoso,DC=com", *wholeSubtree*, "(sAMAccountName=kimakers)", [*objectClass*, *distinguishedName*, *sAMAccountName*, *objectSid*, *sIDHistory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entry:
* >> Dn: CN=Kim Akers,CN=Users,DC=contoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> distinguishedName: CN=Kim Akers,CN=Users,DC=contoso,DC=com;
  + 1> sAMAccountName: KimAkers;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1144;
  + 1> sIDHistory: S-1-5-21-1137440724-3092688314-3181763971-1153;

In the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) DC=LEGACYCONTOSO, DC=COM, the user object whose sAMAccountName is kimakers1 is unchanged:

* ldap\_search\_s("DC=legacycontoso,DC=com", *wholeSubtree*, "(sAMAccountName=kimakers1)", [*objectClass*, *distinguishedName*, *sAMAccountName*, *objectSid*, *sIDHistory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=Kim Akers1,CN=Users,DC=legacycontoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> distinguishedName: CN=Kim Akers1,CN=Users,DC=legacycontoso,DC=com;
  + 1> sAMAccountName: KimAkers1;
  + 1> objectSid: S-1-5-21-1137440724-3092688314-3181763971-1153;

### IDL\_DRSBind (Opnum 0)

The IDL\_DRSBind method creates a context handle that is necessary to call any other method in this interface.

1. ULONG IDL\_DRSBind(
2. [in] handle\_t rpc\_handle,
3. [in, unique] UUID\* puuidClientDsa,
4. [in, unique] DRS\_EXTENSIONS\* pextClient,
5. [out] DRS\_EXTENSIONS\*\* ppextServer,
6. [out, ref] DRS\_HANDLE\* phDrs
7. );

**rpc\_handle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle, as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824).

**puuidClientDsa:** A pointer to a [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) that identifies the caller.

**pextClient:** A pointer to client capabilities, for use in version negotiation.

**ppextServer:** A pointer to a pointer to server capabilities, for use in version negotiation.

**phDrs:** A pointer to an RPC context handle (as specified in [C706]), which can be used in calls to other methods in this interface.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method does not throw exceptions beyond those thrown by the underlying RPC protocol.

#### Client Behavior When Sending the IDL\_DRSBind Request

The client uses *puuidClientDsa* to pass an identifier. If the client uses the returned DRS\_HANDLE for subsequent calls to the [IDL\_DRSWriteSPN](#Section_8b129dc8ed4545379555b6fef764ab7d) method, then the client MUST pass NTDSAPI\_CLIENT\_GUID in *puuidClientDsa*. For any other uses, the server places no constraints on the value of *puuidClientDsa* other than those specified in section [4.1.3.2](#Section_1173CA77049C40FB863CC86ED127AC4E).[<11>](#Appendix_A_11" \o "Product behavior note 11)

The client uses *pextClient* to pass a properly initialized [DRS\_EXTENSIONS\_INT](#Section_3ee529b123db4996948a042f04998e91) structure to the server. If the client is a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), it reads the value of msDS-ReplicationEpoch from its nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) and assigns this value to the **dwReplEpoch** field of the DRS\_EXTENSIONS\_INT structure; otherwise, it sets the **dwReplEpoch** field of the DRS\_EXTENSIONS\_INT structure to zero. If the client is a DC, it reads the value of [**objectGUID**](#gt_ad613dff-e9c4-4cb6-ad6b-0ce52038ceb5) from the Config NC object and assigns this value to the **ConfigObjGUID** field of the DRS\_EXTENSIONS\_INT structure; otherwise, it sets the **ConfigObjGUID** field of the DRS\_EXTENSIONS\_INT structure to the [**NULL GUID**](#gt_ba500a5b-8c29-467c-a335-0980c8b11304) value.

The remaining information in the DRS\_EXTENSIONS\_INT structure must be consistent with the client's capabilities. This information affects the versions of response structures that the server returns in method calls using the [DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) returned by [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d). In descriptions of method calls that use a DRS\_HANDLE, this handle is sometimes called the client's [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context.[<12>](#Appendix_A_12" \o "Product behavior note 12)

If a method of this protocol takes a parameter named *dwInVersion*, the client uses that parameter to specify the version of the referent of the next parameter to that method, often named *pmsgIn*. The referent of this parameter is called the method's request. The *dwInVersion* parameter is called the request version. For example, if the client passes *dwInVersion* = 7 to [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894), the client also passes a [DRS\_MSG\_GETCHGREQ\_V7](#Section_5ef4f597a3974f6fa98b7a034247d886) request.

If a method of this protocol takes an integer parameter named *pdwOutVersion*, the server uses that parameter to return the version number of the referent of the next parameter to that method, often named *pmsgOut*. The referent of this parameter is called the method's response. The referent of *pdwOutVersion* is called the response version. For example, when the server returns *pdwOutVersion*^ = 9 from IDL\_DRSGetNCChanges, the server also returns a [DRS\_MSG\_GETCHGREPLY\_V9](#Section_b9564a194500444ba99b0da1b08cdb6f) response.

Most methods in this protocol are capable of generating only a certain response version from a certain request version. The following special cases apply:

* IDL\_DRSGetNCChanges is capable of returning a version 6 response from version 7, version 8, and version 10 requests. However, the DRS\_EXT\_GETCHGREPLY\_V6 bit must be set in the client's RPC context for the server to generate a version 6 response. Otherwise, the server returns ERROR\_REVISION\_MISMATCH. Note that whenever IDL\_DRSGetNCChanges is capable of returning a version 6 response, it is also capable of returning a version 7 response, which is a compressed form of a version 6 response. Compression of IDL\_DRSGetNCChanges responses is not controlled by the state of the client's RPC context; it is controlled on a per-request basis by the client; see DRS\_USE\_COMPRESSION in section [5.41](#Section_ac9c8a11cd464080acbf9faa86344030).
* IDL\_DRSGetNCChanges is capable of returning a version 9 response from version 10 requests. However, the DRS\_EXT\_GETCHGREPLY\_V9 bit must be set in the client's RPC context for the server to generate a version 9 response. Otherwise, the server returns ERROR\_REVISION\_MISMATCH. Note that whenever IDL\_DRSGetNCChanges is capable of returning a version 9 response, it is also capable of returning a version 7 response, which is a compressed form of a version 9 response. Compression of IDL\_DRSGetNCChanges responses is not controlled by the state of the client's RPC context; it is controlled on a per-request basis by the client; see DRS\_USE\_COMPRESSION in section 5.41.
* [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) can generate either a version 2 or version 3 response from either a version 2 or version 3 request. The server generates a version 3 response when DRS\_EXT\_ADDENTRYREPLY\_V3 is set in the client's RPC context; otherwise, the server generates a version 2 response.
* [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) has only one request version; it contains an **InfoLevel** field. The **InfoLevel**, not the *dwInputVersion*, determines the response version. Similarly, [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47) has two request versions, which both contain an **InfoType** field. The **InfoType**, not the *dwInputVersion*, determines the response version.

The following tables describe how the server determines the response version based on the request version, the DRS\_EXTENSIONS\_INT structure specified when creating the DRS\_HANDLE, and in some cases, the contents of the request message.

[IDL\_DRSReplicaSync](#Section_25c71d91051f4c26977fa70892f29b00)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | - |

IDL\_DRSGetNCChanges

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 4 | - | 1 |
| 5 | - | 1 |
| 7 | DRS\_EXT\_GETCHGREPLY\_V6 | 6 |
| 8 | DRS\_EXT\_GETCHGREPLY\_V6 | 6 |
| 10 | DRS\_EXT\_GETCHGREPLY\_V6 | 6 |
| 10 | DRS\_EXT\_GETCHGREPLY\_V9 | 9 |
| 10 | DRS\_EXT\_GETCHGREPLY\_V9, DRS\_EXT\_GETCHGREPLY\_V6 | 9 |

[IDL\_DRSUpdateRefs](#Section_a273bbcfaeca46088ad4127d3e597cd4)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | - |

[IDL\_DRSReplicaAdd](#Section_7219df914eea494f88e3780d40d2d559)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | - |
| 2 | - | - |

[IDL\_DRSReplicaDel](#Section_1420a9bf9267464da6d57676472d7f1d)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | - |

[IDL\_DRSReplicaModify](#Section_cd241bf256be453786b1cdbc997b0860)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | - |

[IDL\_DRSVerifyNames](#Section_80739a29e8ed44788490475a18e9779d)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSGetMemberships](#Section_d5ace4527cdd4d50bb6439b4c55180a2)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSInterDomainMove](#Section_595b2fef493b4b1db60de7a1a3345b0e)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 2 | - | 2 |

[IDL\_DRSGetNT4ChangeLog](#Section_6e000eb660fd4d6cae82bb6479df02fa)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSCrackNames](#Section_9b4bfb4466564404bcc8dc88111658b3)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

IDL\_DRSWriteSPN

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSRemoveDsServer](#Section_d5c310ae347a49d4a78e6ffb2eecd581)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSRemoveDsDomain](#Section_aa3cfa46c737425aae65ecaf9efe7e84)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

IDL\_DRSDomainControllerInfo

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | request.InfoLevel 1 |

IDL\_DRSAddEntry

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 2 | - | 2 |
| 3 | - | 2 |
| 2 | DRS\_EXT\_ADDENTRYREPLY\_V3 | 3 |
| 3 | DRS\_EXT\_ADDENTRYREPLY\_V3 | 3 |

[IDL\_DRSExecuteKCC](#Section_ad807917687b40d9abe2053af0246523)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

IDL\_DRSGetReplInfo

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | request.InfoType 2 |
| 2 | - | request.InfoType 2 |

[IDL\_DRSAddSidHistory](#Section_376230a5d8064ae5970af6243ee193c8)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSGetMemberships2](#Section_d4e67cc32ee14b2b8055cebefc556252)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSReplicaVerifyObjects](#Section_8dba150d50f647f1941e1a606c30db0b)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | - |

[IDL\_DRSGetObjectExistence](#Section_6355d4f5f5564527adde37afba2fcf56)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSQuerySitesByCost](#Section_2c3faba2d64e4866b8f1fc8d5f4ec710)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSInitDemotion](#Section_faca17da3f7f409298dbfd2ce7d98b8c)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSReplicaDemotion](#Section_8a2f0388bdfb4519a8c3384f27c11639)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSFinishDemotion](#Section_0bf530e81be04f48b8c2208031a8725f)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSAddCloneDC](#Section_ef0bfb1d037b4626a6d9cc7589bc5786)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSWriteNgcKey](#Section_7a140389caa34718bb1ad64483933eb0)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DRSReadNgcKey](#Section_a80c60ac9864444a95136c0c894fbb8d)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DSAPrepareScript](#Section_749197848e574cf5840f6f1bd226cf02)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

[IDL\_DSAExecuteScript](#Section_1cb59761aeae4f448f9e06ae75ae45ca)

| Request version | DRS\_EXTENSIONS bit set in client's RPC context | Response version |
| --- | --- | --- |
| 1 | - | 1 |

1 Possible values are 0x1, 0x2, and 0xffffffff (see section 4.1.5).

2 Possible values are detailed in section 4.1.13.

#### Server Behavior of the IDL\_DRSBind Method

The server returns an error if *puuidClientDsa^* is set to the [**NULL GUID**](#gt_ba500a5b-8c29-467c-a335-0980c8b11304). Otherwise, the server retains the [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) passed as *puuidClientDsa^* and the [DRS\_EXTENSIONS\_INT](#Section_3ee529b123db4996948a042f04998e91) structure passed as *pextClient^*, and associates them with the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle, *phDrs*, in an implementation-specific manner.

The server sets *ppextServer* to a DRS\_EXTENSIONS\_INT structure whose **dwReplEpoch** and **ConfigObjGUID** fields are initialized as described in the previous section ([Client Behavior When Sending the IDL\_DRSBind Request (section 4.1.3.1)](#Section_cbea360497034a0793afd97a1238f3c2)), and whose other fields describe the server.[<13>](#Appendix_A_13" \o "Product behavior note 13) The server associates the information in *ppextServer* with the RPC context handle, *phDrs*, in an implementation-specific manner and then returns a [DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) as the referent of *phDrs*.

The following tables specify the capability assertions made by a server that sets bits in the DRS\_EXTENSIONS\_INT structure returned from [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d). Each row of a table gives a request version (including both *dwInVersion* and the InfoLevel of [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) and the InfoType of [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47)) and the DRS\_EXTENSIONS\_INT bit or bits that the server sets to indicate support for that request. For instance, every server supports a version 1 request to [IDL\_DRSReplicaSync](#Section_25c71d91051f4c26977fa70892f29b00), but a server does not support a version 5 request to [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) unless it has set both the DRS\_EXT\_GETCHGREQ\_V5 and DRS\_EXT\_RESTORE\_USN\_OPTIMIZATION bits. For [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab), the IDL\_DRSDomainControllerInfo method is disabled regardless of the InfoLevel set by the bits.

A server supports version 4 and version 7 requests to IDL\_DRSGetNCChanges only via the SMTP [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) transport (see [[MS-SRPL]](%5bMS-SRPL%5d.pdf#Section_ec69eea50d5e428ab5bc66732aaeb866)). These cases are noted in the relevant table. A server supports all other requests only via the [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a).

IDL\_DRSReplicaSync

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | - |

IDL\_DRSGetNCChanges

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 4 | SMTP replication transport |
| 5 | DRS\_EXT\_GETCHGREQ\_V5  DRS\_EXT\_RESTORE\_USN\_OPTIMIZATION |
| 7 | SMTP replication transport |
| 8 | DRS\_EXT\_GETCHGREQ\_V8  DRS\_EXT\_RESTORE\_USN\_OPTIMIZATION |
| 10 | DRS\_EXT\_GETCHGREQ\_V10  DRS\_EXT\_RESTORE\_USN\_OPTIMIZATION |

[IDL\_DRSUpdateRefs](#Section_a273bbcfaeca46088ad4127d3e597cd4)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | - |

[IDL\_DRSReplicaAdd](#Section_7219df914eea494f88e3780d40d2d559)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | - |
| 2 | DRS\_EXT\_ASYNCREPL |

[IDL\_DRSReplicaDel](#Section_1420a9bf9267464da6d57676472d7f1d)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | - |

[IDL\_DRSReplicaModify](#Section_cd241bf256be453786b1cdbc997b0860)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | - |

[IDL\_DRSVerifyNames](#Section_80739a29e8ed44788490475a18e9779d)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | - |

[IDL\_DRSGetMemberships](#Section_d5ace4527cdd4d50bb6439b4c55180a2)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | - |

[IDL\_DRSInterDomainMove](#Section_595b2fef493b4b1db60de7a1a3345b0e)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 2 | DRS\_EXT\_MOVEREQ\_V2 |

[IDL\_DRSGetNT4ChangeLog](#Section_6e000eb660fd4d6cae82bb6479df02fa)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | - |

[IDL\_DRSCrackNames](#Section_9b4bfb4466564404bcc8dc88111658b3)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | - |

[IDL\_DRSWriteSPN](#Section_8b129dc8ed4545379555b6fef764ab7d)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | - |

[IDL\_DRSRemoveDsServer](#Section_d5c310ae347a49d4a78e6ffb2eecd581)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | DRS\_EXT\_REMOVEAPI |

[IDL\_DRSRemoveDsDomain](#Section_aa3cfa46c737425aae65ecaf9efe7e84)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | DRS\_EXT\_REMOVEAPI |

IDL\_DRSDomainControllerInfo

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1  InfoLevel = 0x1 | DRS\_EXT\_DCINFO\_V1 |
| 1  InfoLevel = 0x2 | DRS\_EXT\_DCINFO\_V2 |
| 1  InfoLevel = 0x3 | DRS\_EXT\_LH\_BETA2 |
| 1  InfoLevel = 0xffffffff | DRS\_EXT\_DCINFO\_VFFFFFFFF |

[IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 2 | DRS\_EXT\_ADDENTRY\_V2 |
| 3 | DRS\_EXT\_NONDOMAIN\_NCS |

[IDL\_DRSExecuteKCC](#Section_ad807917687b40d9abe2053af0246523)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | DRS\_EXT\_KCC\_EXECUTE |

IDL\_DRSGetReplInfo

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | - |
| 2 | DRS\_EXT\_GETCHGREQ\_V8 |
| 2  InfoType = [3..5] | DRS\_EXT\_POST\_BETA3 |
| 2  InfoType = 6 | DRS\_EXT\_GETCHGREQ\_V8 |
| 2  InfoType = [7..10] | DRS\_EXT\_GETCHGREPLY\_V6 |

[IDL\_DRSAddSidHistory](#Section_376230a5d8064ae5970af6243ee193c8)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | DRS\_EXT\_ADD\_SID\_HISTORY |

[IDL\_DRSGetMemberships2](#Section_d4e67cc32ee14b2b8055cebefc556252)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | DRS\_EXT\_GETMEMBERSHIPS2 |

[IDL\_DRSReplicaVerifyObjects](#Section_8dba150d50f647f1941e1a606c30db0b)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | DRS\_EXT\_WHISTLER\_BETA3 |

[IDL\_DRSGetObjectExistence](#Section_6355d4f5f5564527adde37afba2fcf56)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | DRS\_EXT\_WHISTLER\_BETA3 |

[IDL\_DRSQuerySitesByCost](#Section_2c3faba2d64e4866b8f1fc8d5f4ec710)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | DRS\_EXT\_WHISTLER\_BETA3 |

[IDL\_DRSInitDemotion](#Section_faca17da3f7f409298dbfd2ce7d98b8c)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | DRS\_EXT\_ADAM |

[IDL\_DRSReplicaDemotion](#Section_8a2f0388bdfb4519a8c3384f27c11639)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | DRS\_EXT\_ADAM |

[IDL\_DRSFinishDemotion](#Section_0bf530e81be04f48b8c2208031a8725f)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | DRS\_EXT\_ADAM |

[IDL\_DSAPrepareScript](#Section_749197848e574cf5840f6f1bd226cf02)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | - |

[IDL\_DSAExecuteScript](#Section_1cb59761aeae4f448f9e06ae75ae45ca)

| Request version | DRS\_EXTENSIONS\_INT bit(s) |
| --- | --- |
| 1 | - |

#### Client Behavior When Receiving the IDL\_DRSBind Response

The client receives a [DRS\_EXTENSIONS\_INT](#Section_3ee529b123db4996948a042f04998e91) structure from the server as the referent of *ppextServer*.

A server supports only a subset of the possible request versions, including both *dwInVersion* and the InfoLevel of [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) and the InfoType of [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47). The server informs the client of its capabilities via the DRS\_EXTENSIONS\_INT structure returned from [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d), as described in [Server Behavior of the IDL\_DRSBind Method (section 4.1.3.2)](#Section_1173ca77049c40fb863cc86ed127ac4e).

The client receives a [DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) as the referent of *phDrs*.

The client retains the context handle *phDrs^* for use in method calls on the drsuapi interface. Once a valid handle has been acquired by the client, the handle remains valid until either the server unilaterally breaks the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) connection (for example, by crashing) or until [IDL\_DRSUnbind](#Section_49eb17c9b6a94ceabef866abda8a7850) has been performed.

#### Examples of the IDL\_DRSBind Method

The [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) Server on DC2.CONTOSO.COM is binding to the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) server DC1.CONTOSO.COM.

##### Initial State

Querying the nTDSDSA [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for the root domain NC DC=CONTOSO, DC=COM for DC1 and DC2 respectively:

* ldap\_search\_s("CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com", *baseObject*, "(objectClass=\*)", [*objectClass, cn, distinguishedName, objectGUID, msDS-Behavior-Version*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com

3> objectClass: top; applicationSettings; nTDSDSA;

1> cn: NTDS Settings;

1> distinguishedName: CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com;

1> objectGUID: c20bc312-4d35-4cc0-9903-b1073368af4a;

1> msDS-Behavior-Version: 2 = (DS\_BEHAVIOR\_WIN2003);

* ldap\_search\_s("CN=NTDS Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com", *baseObject*, "(objectClass=\*)", [*objectClass, cn, distinguishedName, objectGUID, msDS-Behavior-Version*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=NTDS Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com

3> objectClass: top; applicationSettings; nTDSDSA;

1> cn: NTDS Settings;

1> distinguishedName: CN=NTDS Settings, CN=DC2, CN=Servers, CN=Default-First-Site-Name, CN=Sites, CN=Configuration, DC=contoso, DC=com;

1> objectGUID: 6aad8f5a-07cc-403a-9696-9102fe1c320b;

1> msDS-Behavior-Version: 2 = (DS\_BEHAVIOR\_WIN2003)

##### Client Request

DC2 invokes the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method against DC1, with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *puuidClientDsa* = GUID {6aad8f5a-07cc-403a-9696-9102fe1c320b}
* *pextClient*:
  + cb: 0x30
  + dwFlags:
    - DRS\_EXT\_BASE
    - DRS\_EXT\_ASYNCREPL
    - DRS\_EXT\_REMOVEAPI
    - DRS\_EXT\_MOVEREQ\_V2
    - DRS\_EXT\_GETCHG\_DEFLATE
    - DRS\_EXT\_DCINFO\_V1
    - DRS\_EXT\_RESTORE\_USN\_OPTIMIZATION
    - DRS\_EXT\_KCC\_EXECUTE
    - DRS\_EXT\_ADDENTRY\_V2
    - DRS\_EXT\_LINKED\_VALUE\_REPLICATION
    - DRS\_EXT\_DCINFO\_V2
    - DRS\_EXT\_INSTANCE\_TYPE\_NOT\_REQ\_ON\_MOD
    - DRS\_EXT\_CRYPTO\_BIND
    - DRS\_EXT\_GET\_REPL\_INFO
    - DRS\_EXT\_STRONG\_ENCRYPTION
    - DRS\_EXT\_DCINFO\_VFFFFFFFF
    - DRS\_EXT\_TRANSITIVE\_MEMBERSHIP
    - DRS\_EXT\_ADD\_SID\_HISTORY
    - DRS\_EXT\_POST\_BETA3
    - DRS\_EXT\_GETCHGREQ\_V5
    - DRS\_EXT\_GET\_MEMBERSHIPS2
    - DRS\_EXT\_GETCHGREQ\_V6
    - DRS\_EXT\_NONDOMAIN\_NCS
    - DRS\_EXT\_GETCHGREQ\_V8
    - DRS\_EXT\_GETCHGREPLY\_V5
    - DRS\_EXT\_GETCHGREPLY\_V6
    - DRS\_EXT\_GETCHGREPLY\_V9
    - DRS\_EXT\_WHISTLER\_BETA3
    - DRS\_EXT\_W2K3\_DEFLATE
    - DRS\_EXT\_GETCHGREQ\_V10
  + SiteObjGuid: [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) {620954c7-7044-400f-9c0b-5c9154198aa6}
  + Pid: 632
  + dwReplEpoch:0
  + dwFlagsExt: 0
  + ConfigObjGUID: [**NULL GUID**](#gt_ba500a5b-8c29-467c-a335-0980c8b11304)

##### Server Response

Return code of 0 ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted) with the following values:

* *ppextServer*:
  + cb: 0x30
  + dwFlags:
    - DRS\_EXT\_BASE
    - DRS\_EXT\_ASYNCREPL
    - DRS\_EXT\_REMOVEAPI
    - DRS\_EXT\_MOVEREQ\_V2
    - DRS\_EXT\_GETCHG\_DEFLATE
    - DRS\_EXT\_DCINFO\_V1
    - DRS\_EXT\_RESTORE\_USN\_OPTIMIZATION
    - DRS\_EXT\_KCC\_EXECUTE
    - DRS\_EXT\_ADDENTRY\_V2
    - DRS\_EXT\_LINKED\_VALUE\_REPLICATION
    - DRS\_EXT\_DCINFO\_V2
    - DRS\_EXT\_INSTANCE\_TYPE\_NOT\_REQ\_ON\_MOD
    - DRS\_EXT\_GET\_REPL\_INFO
    - DRS\_EXT\_STRONG\_ENCRYPTION
    - DRS\_EXT\_DCINFO\_VFFFFFFFF
    - DRS\_EXT\_TRANSITIVE\_MEMBERSHIP
    - DRS\_EXT\_ADD\_SID\_HISTORY
    - DRS\_EXT\_POST\_BETA3
    - DRS\_EXT\_GETCHGREQ\_V5
    - DRS\_EXT\_GET\_MEMBERSHIPS2
    - DRS\_EXT\_GETCHGREQ\_V6
    - DRS\_EXT\_NONDOMAIN\_NCS
    - DRS\_EXT\_GETCHGREQ\_V8
    - DRS\_EXT\_GETCHGREPLY\_V5
    - DRS\_EXT\_GETCHGREPLY\_V6
    - DRS\_EXT\_GETCHGREPLY\_V9
    - DRS\_EXT\_WHISTLER\_BETA3
    - DRS\_EXT\_W2K3\_DEFLATE
    - DRS\_EXT\_GETCHGREQ\_V10
  + SiteObjGuid: [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) {620954c7-7044-400f-9c0b-5c9154198aa6}
  + Pid: 632
  + dwReplEpoch: 0
  + dwFlagsExt: 0
  + ConfigObjGUID: [**NULL GUID**](#gt_ba500a5b-8c29-467c-a335-0980c8b11304)

##### Final State

No change in state.

### IDL\_DRSCrackNames (Opnum 12)

The IDL\_DRSCrackNames method looks up each of a set of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) and returns it to the caller in the requested format.

1. ULONG IDL\_DRSCrackNames(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_CRACKREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_CRACKREPLY\* pmsgOut
9. );

**hDrs:** [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** Version of the request message.

**pmsgIn:** Pointer to the request message.

**pdwOutVersion:** Pointer to the version of the response message.

**pmsgOut:** Pointer to the response message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_CRACKREQ

The DRS\_MSG\_CRACKREQ union defines the request messages sent to the [IDL\_DRSCrackNames](#Section_9b4bfb4466564404bcc8dc88111658b3) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_CRACKREQ\_V1 V1;
6. } DRS\_MSG\_CRACKREQ;

**V1:**  Version 1 request.

##### DRS\_MSG\_CRACKREQ\_V1

The DRS\_MSG\_CRACKREQ\_V1 structure defines the request message sent to the [IDL\_DRSCrackNames](#Section_9b4bfb4466564404bcc8dc88111658b3) method.

1. typedef struct {
2. ULONG CodePage;
3. ULONG LocaleId;
4. DWORD dwFlags;
5. DWORD formatOffered;
6. DWORD formatDesired;
7. [range(1,10000)] DWORD cNames;
8. [string, size\_is(cNames)] WCHAR\*\* rpNames;
9. } DRS\_MSG\_CRACKREQ\_V1;

**CodePage:**  The character set used by the client. This field SHOULD be ignored by the server.

**LocaleId:**  The locale used by the client. This field SHOULD be ignored by the server.

**dwFlags:**  Zero or more of the following bit flags, which are presented in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | T R | G C | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | F P O | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**GC (DS\_NAME\_FLAG\_GCVERIFY, 0x00000004)**: If set, the call fails if the server is not a [**GC server**](#gt_a5a99ce4-e206-42dc-8874-e103934c5b0d).

**TR (DS\_NAME\_FLAG\_TRUST\_REFERRAL, 0x00000008)**: If set and the lookup fails on the server, referrals are returned to trusted [**forests**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) where the lookup might succeed.

**FPO (DS\_NAME\_FLAG\_PRIVATE\_RESOLVE\_FPOS, 0x80000000)**: If set and the named [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) is a foreign [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409), indicate this by using the status of the lookup operation.

**formatOffered:**  The format of the names in **rpNames**. This can be one of the values from [DS\_NAME\_FORMAT (section 4.1.4.1.3)](#Section_73c73cf208244d6597f4f56244f3e8a6) or one of the following.

| Value | Meaning |
| --- | --- |
| DS\_LIST\_SITES  0xFFFFFFFF | Get all [**sites**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba) in the forest. |
| DS\_LIST\_SERVERS\_IN\_SITE  0xFFFFFFFE | Get all servers in a given site. |
| DS\_LIST\_DOMAINS\_IN\_SITE  0xFFFFFFFD | Get all [**domains**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) in a given site. |
| DS\_LIST\_SERVERS\_FOR\_DOMAIN\_IN\_SITE  0xFFFFFFFC | Get all [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) of a specified domain in a given site. |
| DS\_LIST\_INFO\_FOR\_SERVER  0xFFFFFFFB | Get DNS host name and server reference for a given DC. |
| DS\_LIST\_ROLES  0xFFFFFFFA | Get [**FSMO role owners**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b). |
| DS\_NT4\_ACCOUNT\_NAME\_SANS\_DOMAIN  0xFFFFFFF9 | Get value of sAMAccountName [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f). |
| DS\_MAP\_SCHEMA\_GUID  0xFFFFFFF8 | Get [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) display name from [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093) [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1). The given schema GUID must be in the curly braced GUID string format as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.3.4.3. |
| DS\_LIST\_DOMAINS  0xFFFFFFF7 | Get all domains in the forest. |
| DS\_LIST\_NCS  0xFFFFFFF6 | Get all [**NCs**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) in the forest. |
| DS\_ALT\_SECURITY\_IDENTITIES\_NAME  0xFFFFFFF5 | Compares input names against the values of the **altSecurityIdentities** attribute. |
| DS\_STRING\_SID\_NAME  0xFFFFFFF4 | String form of [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d). |
| DS\_LIST\_SERVERS\_WITH\_DCS\_IN\_SITE  0xFFFFFFF3 | Get all DCs in a given site. |
| DS\_LIST\_GLOBAL\_CATALOG\_SERVERS  0xFFFFFFF1 | Get all [**GCs**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169) in the forest. |
| DS\_NT4\_ACCOUNT\_NAME\_SANS\_DOMAIN\_EX  0xFFFFFFF0 | Get value of sAMAccountName attribute; return status DS\_NAME\_ERROR\_NOT\_FOUND if account is invalid. |
| DS\_USER\_PRINCIPAL\_NAME\_AND\_ALTSECID  0xFFFFFFEF | Compares input names against the user [**principal**](#gt_8492780e-99e2-47ba-8553-aedb8de9f9c0) name and the values of the **altSecurityIdentities** attribute. |

**formatDesired:**  Format of the names in the **rItems** field of the [DS\_NAME\_RESULTW](#Section_0076d2413f794b0b8e078ccfaff8bd4c) structure, which is returned inside the [DRS\_MSG\_CRACKREPLY](#Section_1dc605fedd85481d84a4f4c5da812d57) message. This can be one of the values from DS\_NAME\_FORMAT or one of the following.

| Value | Meaning |
| --- | --- |
| DS\_STRING\_SID\_NAME  0xFFFFFFF4 | String form of a SID. |
| DS\_USER\_PRINCIPAL\_NAME\_FOR\_LOGON  0xFFFFFFF2 | User principal name. |

**cNames:**  Count of items in the **rpNames** array.

**rpNames:**  Input names to translate.

##### DS\_NAME\_FORMAT

The DS\_NAME\_FORMAT enumeration describes the format of a name sent to or received from the [IDL\_DRSCrackNames](#Section_9b4bfb4466564404bcc8dc88111658b3) method.

1. typedef enum
2. {
3. DS\_UNKNOWN\_NAME = 0,
4. DS\_FQDN\_1779\_NAME = 1,
5. DS\_NT4\_ACCOUNT\_NAME = 2,
6. DS\_DISPLAY\_NAME = 3,
7. DS\_UNIQUE\_ID\_NAME = 6,
8. DS\_CANONICAL\_NAME = 7,
9. DS\_USER\_PRINCIPAL\_NAME = 8,
10. DS\_CANONICAL\_NAME\_EX = 9,
11. DS\_SERVICE\_PRINCIPAL\_NAME = 10,
12. DS\_SID\_OR\_SID\_HISTORY\_NAME = 11,
13. DS\_DNS\_DOMAIN\_NAME = 12
14. } DS\_NAME\_FORMAT;

**DS\_UNKNOWN\_NAME:** The server looks up the name by using the algorithm specified in the LookupUnknownName procedure.

**DS\_FQDN\_1779\_NAME:** A [**distinguished name**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b).

**DS\_NT4\_ACCOUNT\_NAME:** Windows NT 4.0 (and prior) name format. The account name is in the format domain\user and the domain-only name is in the format domain\.

**DS\_DISPLAY\_NAME:** A user-friendly display name.

**DS\_UNIQUE\_ID\_NAME:** Curly braced string representation of an objectGUID. The format of the string representation is specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.3.4.3.

**DS\_CANONICAL\_NAME:** A [**canonical name**](#gt_79ab9d86-0d30-41c3-b7da-153ad41bdfd8).

**DS\_USER\_PRINCIPAL\_NAME:** User [**principal**](#gt_8492780e-99e2-47ba-8553-aedb8de9f9c0) name.

**DS\_CANONICAL\_NAME\_EX:** Same as DS\_CANONICAL\_NAME except that the rightmost forward slash (/) is replaced with a newline character (\n).

**DS\_SERVICE\_PRINCIPAL\_NAME:** [**Service principal name (SPN)**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4).

**DS\_SID\_OR\_SID\_HISTORY\_NAME:** String representation of a [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) (as specified in [MS-DTYP] section 2.4.2).

**DS\_DNS\_DOMAIN\_NAME:** Not supported.

##### DS\_NAME\_RESULT\_ITEMW

The DS\_NAME\_RESULT\_ITEMW structure defines the translated name returned by the [IDL\_DRSCrackNames](#Section_9b4bfb4466564404bcc8dc88111658b3) method.

1. typedef struct {
2. DWORD status;
3. [string, unique] WCHAR\* pDomain;
4. [string, unique] WCHAR\* pName;
5. } DS\_NAME\_RESULT\_ITEMW,
6. \*PDS\_NAME\_RESULT\_ITEMW;

**status:**  Status of the crack name operation for the corresponding element of the **rpNames** field in the request. The status is one of the values from the enumeration [DS\_NAME\_ERROR](#Section_7f70adbb159844f99df15c89ca87225f).

**pDomain:**  DNS [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) name of the domain in which the named [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) resides.

**pName:**  Object name in the requested format.

##### DS\_NAME\_RESULTW

The DS\_NAME\_RESULTW structure defines the translated names returned by the [IDL\_DRSCrackNames](#Section_9b4bfb4466564404bcc8dc88111658b3) method.

1. typedef struct {
2. DWORD cItems;
3. [size\_is(cItems)] PDS\_NAME\_RESULT\_ITEMW rItems;
4. } DS\_NAME\_RESULTW,
5. \*PDS\_NAME\_RESULTW;

**cItems:**  The count of items in the **rItems** array.

**rItems:**  Translated names that correspond one-to-one with the elements in the **rpNames** field of the request.

##### DRS\_MSG\_CRACKREPLY

The DRS\_MSG\_CRACKREPLY union defines the response messages received from the [IDL\_DRSCrackNames](#Section_9b4bfb4466564404bcc8dc88111658b3) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_CRACKREPLY\_V1 V1;
6. } DRS\_MSG\_CRACKREPLY;

**V1:**  Version 1 reply.

##### DRS\_MSG\_CRACKREPLY\_V1

The DRS\_MSG\_CRACKREPLY\_V1 structure defines the response message received from the [IDL\_DRSCrackNames](#Section_9b4bfb4466564404bcc8dc88111658b3) method.

1. typedef struct {
2. DS\_NAME\_RESULTW\* pResult;
3. } DRS\_MSG\_CRACKREPLY\_V1;

**pResult:**  Translated form of the names.

##### DS\_NAME\_ERROR

This section enumerates the possible statuses of a translation operation.

| Symbolic name | Description |
| --- | --- |
| 0  DS\_NAME\_NO\_ERROR | No error occurred during the name translation. |
| 1  DS\_NAME\_ERROR\_RESOLVING | Generic processing error during the name translation. |
| 2  DS\_NAME\_ERROR\_NOT\_FOUND | The [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) with the specified name cannot be found. |
| 3  DS\_NAME\_ERROR\_NOT\_UNIQUE | More than one object is located with the specified name. |
| 4  DS\_NAME\_ERROR\_NO\_MAPPING | The desired output format cannot be applied to the object with the specified name. |
| 5  DS\_NAME\_ERROR\_DOMAIN\_ONLY | Only the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) part of the name was translated. |
| 7  DS\_NAME\_ERROR\_TRUST\_REFERRAL | The specified name belongs to a trusted [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62), a referral is returned. |
| 0xFFFFFFF2  DS\_NAME\_ERROR\_IS\_SID\_HISTORY\_UNKNOWN | The specified name matches a value in the **sidHistory** [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of an object, but the type of the object is unknown. |
| 0xFFFFFFF3  DS\_NAME\_ERROR\_IS\_SID\_HISTORY\_ALIAS | Translation was successful. The specified name matches a value in the **sidHistory** attribute of an object. The object's **sAMAccountType** attribute value is either SAM\_NON\_SECURITY\_ALIAS\_OBJECT or SAM\_ALIAS\_OBJECT as defined in [[MS-SAMR]](%5bMS-SAMR%5d.pdf#Section_4df07fab1bbc452f8e927853a3c7e380) section 2.2.1.9, ACCOUNT\_TYPE Values. |
| 0xFFFFFFF4  DS\_NAME\_ERROR\_IS\_SID\_HISTORY\_GROUP | Translation was successful. The specified name matches a value in the **sidHistory** attribute of an object. The object's **sAMAccountType** attribute value is either SAM\_GROUP\_OBJECT or SAM\_NON\_SECURITY\_GROUP\_OBJECT as defined in [MS-SAMR] section 2.2.1.9, ACCOUNT\_TYPE Values. |
| 0xFFFFFFF5  DS\_NAME\_ERROR\_IS\_SID\_HISTORY\_USER | Translation was successful. The specified name matches a value in the **sidHistory** attribute of an object. The object's **sAMAccountType** attribute value is SAM\_USER\_OBJECT or SAM\_MACHINE\_ACCOUNT or SAM\_TRUST\_ACCOUNT as defined in [MS-SAMR] section 2.2.1.9, ACCOUNT\_TYPE Values. |
| 0xFFFFFFF6  DS\_NAME\_ERROR\_IS\_SID\_UNKNOWN | The specified name matches the [**objectSid**](#gt_bc1ccb71-6369-44b1-837b-ec503e94baca) attribute of an object, but the type of the object is unknown. |
| 0xFFFFFFF7  DS\_NAME\_ERROR\_IS\_SID\_ALIAS | Translation was successful. The specified name matches the objectSid attribute of an object. The object's **sAMAccountType** attribute value is either SAM\_NON\_SECURITY\_ALIAS\_OBJECT or SAM\_ALIAS\_OBJECT as defined in [MS-SAMR] section 2.2.1.9, ACCOUNT\_TYPE Values. |
| 0xFFFFFFF8  DS\_NAME\_ERROR\_IS\_SID\_GROUP | Translation was successful. The specified name matches the objectSid attribute of an object. The object's **sAMAccountType** attribute value is either SAM\_GROUP\_OBJECT or SAM\_NON\_SECURITY\_GROUP\_OBJECT as defined in [MS-SAMR] section 2.2.1.9, ACCOUNT\_TYPE Values. |
| 0xFFFFFFF9  DS\_NAME\_ERROR\_IS\_SID\_USER | Translation was successful. The specified name matches the objectSid attribute of an object. The object's **sAMAccountType** attribute value is SAM\_USER\_OBJECT or SAM\_MACHINE\_ACCOUNT or SAM\_TRUST\_ACCOUNT as defined in [MS-SAMR] section 2.2.1.9, ACCOUNT\_TYPE Values. |
| 0xFFFFFFFA  DS\_NAME\_ERROR\_SCHEMA\_GUID\_CONTROL\_RIGHT | Translation was successful. The [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) identifies a [**control access right**](#gt_42f6c9e0-a2b3-4bc3-9b87-fdb902e5505e). |
| 0xFFFFFFFB  DS\_NAME\_ERROR\_SCHEMA\_GUID\_CLASS | Translation was successful. The GUID identifies a classSchema object. |
| 0xFFFFFFFC  DS\_NAME\_ERROR\_SCHEMA\_GUID\_ATTR\_SET | Translation was successful. The GUID identifies a property set. |
| 0xFFFFFFFD  DS\_NAME\_ERROR\_SCHEMA\_GUID\_ATTR | Translation was successful. The GUID identifies an attributeSchema object. |
| 0xFFFFFFFE  DS\_NAME\_ERROR\_SCHEMA\_GUID\_NOT\_FOUND | The GUID cannot be resolved. |
| 0xFFFFFFFF  DS\_NAME\_ERROR\_IS\_FPO | The object with the specified name is a Foreign Principal Object. |

#### Method-Specific Abstract Types and Procedures

##### CanonicalNameFromCanonicalNameEx

1. procedure CanonicalNameFromCanonicalNameEx(
2. name: unicodestring): unicodestring

This procedure converts *name* from [**extended canonical name**](#gt_4647b948-36f2-4ff2-a86c-2c6ef3c45b8d) format to [**canonical name**](#gt_79ab9d86-0d30-41c3-b7da-153ad41bdfd8) format by replacing the last newline character in *name* with a forward slash character. If *name* is not in the correct format, "domain/container/container/.../container\nleaf" (where \n designates a newline character), this procedure returns null.

##### DomainDNSNameFromDomain

1. procedure DomainDNSNameFromDomain(domainNC: DSName): unicodestring

If the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef), whose root has the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) *domainNC*, is hosted in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62), this procedure returns the DNS [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) name of that domain NC. Otherwise, null is returned.

##### DomainFromDomainDNSName

1. procedure DomainFromDomainDNSName(domainName: unicodestring): DSName

If the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) hosts an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) of the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) whose DNS [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) name is *domainName*, this procedure returns the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the root of that domain NC. Otherwise, it returns null.

##### DomainNameFromCanonicalName

1. procedure DomainNameFromCanonicalName(
2. canonicalName: unicodestring): unicodestring

Given a name in canonical format, this procedure extracts and returns the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) [**FQDN (1)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e). If the input is not in [**canonical name**](#gt_79ab9d86-0d30-41c3-b7da-153ad41bdfd8) format, then null is returned. For example, when the input is "example.fabrikam.com/container/username", the returned domain FQDN (1) is "example.fabrikam.com".

##### DomainNameFromSid

1. procedure DomainNameFromSid(domainSid: SID): unicodestring

Looks up the [**domain SID**](#gt_c1d6ba4d-2302-43a5-acd2-02bfe19d0ade) *domainSid* among trusted [**domains**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) and domains in trusted [**forests**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). If *domainSid* is the domain SID of a trusted domain, then the name of this domain is returned. If the input is null, then null is returned.

##### DomainNameFromUPN

1. procedure DomainNameFromUPN(upn: unicodestring): unicodestring

Parses and returns the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) name from a UPN-formatted string *upn*. The domain name is the component after the '@'. For example, when the input is "username@example.fabrikam.com", then "example. fabrikam.com" is returned. If *upn* is not in UPN format, then null is returned.

##### DomainNetBIOSNameFromDomain

1. procedure DomainNetBIOSNameFromDomain(domainNC: DSName): unicodestring

If the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef), whose root has the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) *domainNC*, is hosted in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62), this procedure returns the NetBIOS [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) name of that domain NC. Otherwise, null is returned.

##### DomainSidFromSid

1. procedure DomainSidFromSid(sid: SID): SID

Removes the last subauthority from the input [**security identifier**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) *sid* and returns the resulting security identifier, which is the [**domain SID**](#gt_c1d6ba4d-2302-43a5-acd2-02bfe19d0ade). If the input is null, the procedure returns null. See [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2 for more details on SIDs.

##### CrackNames

1. procedure CrackNames(DRS\_MSG\_CRACKREQ\_V1 msgIn, DS\_NAME\_RESULTW \*pmsgOut): ULONG

The CrackNames method implements the core functionality of [IDL\_DRSCrackNames](#Section_9b4bfb4466564404bcc8dc88111658b3), that is, looking up [**directory object**](#gt_407dbc2c-3140-4e31-9085-0087e2d3bab2) names that are provided in one format (for example, [**SPNs**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4)) and returning them in a different format (for example, [**DNs**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b)).

1. i: DWORD
2. rt: set of DSName
3. serverObj, siteObj, attr, class, er: DSName
4. guid: GUID
5. if msgIn.formatOffered in {
6. all constants in DS\_NAME\_FORMAT enumeration,
7. DS\_NT4\_ACCOUNT\_NAME\_SANS\_DOMAIN,
8. DS\_NT4\_ACCOUNT\_NAME\_SANS\_DOMAIN\_EX,
9. DS\_ALT\_SECURITY\_IDENTITIES\_NAME,
10. DS\_STRING\_SID\_NAME,
11. DS\_USER\_PRINCIPAL\_NAME\_AND\_ALTSECID} then
12. /\* Regular name lookup. \*/
13. for i := 0 to msgIn.cNames - 1
14. /\* Perform the lookup based on the input format. \*/
15. msgOut^.rItems[i] := LookupName(
16. msgIn.dwFlags, msgIn.formatOffered, msgIn.formatDesired,
17. msgIn.rpNames[i])
18. endfor
19. msgOut^.cItems = msgIn.cNames
20. else if msgIn.formatOffered = DS\_LIST\_ROLES then
21. /\* Return the list of FSMO role owners. \*/
22. i := 0
23. foreach role in {FSMO\_SCHEMA, FSMO\_DOMAIN\_NAMING, FSMO\_PDC,
24. FSMO\_RID, FSMO\_INFRASTRUCTURE}
25. msgOut^.rItems[i].pName := GetFSMORoleOwner(role).dn
26. msgOut^.rItems[i].status := DS\_NAME\_NO\_ERROR
27. i := i + 1
28. endfor
29. msgOut^.cItems := i
30. else if msgIn.formatOffered = DS\_LIST\_SITES then
31. /\* Return the list of known sites. \*/
32. rt := select all o from children
33. DescendantObject(ConfigNC(),"CN=Sites,")
34. where o!objectCategory = GetDefaultObjectCategory(site)
35. i := 0
36. foreach siteObj in rt
37. msgOut^.rItems[i].pName := siteObj.dn
38. msgOut^.rItems[i].status := DS\_NAME\_NO\_ERROR
39. i := i + 1
40. endfor
41. msgOut^.cItems := i
42. else if msgIn.formatOffered = DS\_LIST\_SERVERS\_IN\_SITE then
43. /\* Return all DCs in a site named msgIn.rpNames[0]. \*/
44. rt := select all o from subtree msgIn.rpNames[0]
45. where o!objectCategory = GetDefaultObjectCategory(server)
46. i := 0
47. foreach serverObj in rt
48. msgOut^.rItems[i].pName := serverObj.dn
49. msgOut^.rItems[i].status := DS\_NAME\_NO\_ERROR
50. i := i + 1
51. endfor
52. msgOut^.cItems := i
53. else if msgIn.formatOffered = DS\_LIST\_DOMAINS then
54. /\* Return all known AD domains. \*/
55. rt := select all o from
56. subtree DescendantObject(ConfigNC(), "CN=Partitions,")
57. where o!objectCategory = GetDefaultObjectCategory(crossRef)
58. and FLAG\_CR\_NTDS\_DOMAIN in o!systemFlags
59. i := 0
60. foreach crObj in rt
61. msgOut^.rItems[i].pName := crObj!ncName.dn
62. msgOut^.rItems[i].status := DS\_NAME\_NO\_ERROR
63. i := i + 1
64. endfor
65. msgOut^.cItems := i
66. else if msgIn.formatOffered = DS\_LIST\_NCS then
67. /\* Return all known NCs. \*/
68. rt := select all o from
69. subtree DescendantObject(ConfigNC(), "CN=Partitions,")
70. where o!objectCategory = GetDefaultObjectCategory(crossRef)
71. i := 0
72. foreach crObj in rt
73. msgOut^.rItems[i].pName := crObj!ncName.dn
74. msgOut^.rItems[i].status := DS\_NAME\_NO\_ERROR
75. i := i + 1
76. endfor
77. msgOut^.cItems := i
78. else if msgIn.formatOffered = DS\_LIST\_DOMAINS\_IN\_SITE then
79. /\* Return the list of domains that are hosted by DCs in a site
80. \* named msgIn.rpNames[0]. \*/
81. /\* First find all DCs in a site named msgIn.rpNames[0]. \*/
82. rt := select all o from subtree msgIn.rpNames[0]
83. where o!objectCategory = GetDefaultObjectCategory(nTDSDSA)
84. /\* Gather the list of all domains from DSA object. \*/
85. hostedDomains := null
86. foreach dsaObj in rt
87. /\* Union operation eliminates duplicates. \*/
88. hostedDomains := hostedDomains + dsaObj!hasMasterNCs
89. endfor
90. i := 0
91. foreach domain in hostedDomains
92. if domain ≠ SchemaNC() and domain ≠ ConfigNC() then
93. msgOut^.rItems[i].pName := domain.dn
94. msgOut^.rItems[i].status := DS\_NAME\_NO\_ERROR
95. i := i + 1
96. endif
97. endfor
98. msgOut^.cItems := i
99. else if msgIn.formatOffered = DS\_LIST\_SERVERS\_FOR\_DOMAIN\_IN\_SITE then
100. /\* Return all DSAs hosting domain msgIn.rpNames[0] in a site named
101. \* msgIn.rpNames[1]. \*/
102. rt := select all o from subtree msgIn.rpNames[1]
103. where o!objectCategory = GetDefaultObjectCategory(nTDSDSA)
104. and msgIn.rpNames[0] in o!msDS-hasMasterNCs
105. /\* Return the list of server objects (parents of DSAs). \*/
106. i := 0
107. foreach dsaObj in rt
108. serverObj := select one o from subtree ConfigNC() where
109. o!objectGUID = dsaObj!parent
110. msgOut^.rItems[i].pName := serverObj.dn
111. msgOut^.rItems[i].status := DS\_NAME\_NO\_ERROR
112. i := i + 1
113. endfor
114. msgOut^.cItems := i
115. else if msgIn.formatOffered = DS\_LIST\_SERVERS\_WITH\_DCS\_IN\_SITE then
116. /\* Return all servers that have DSA objects in a site named
117. \* msgIn.rpNames[0]. \*/
118. rt := select all o from subtree msgIn.rpNames[0]
119. where o!objectCategory = GetDefaultObjectCategory(nTDSDSA)
120. and o!hasMasterNCs ≠ null
121. /\* Return the list of server objects (parents of DSAs). \*/
122. i := 0
123. foreach dsaObj in rt
124. serverObj := select one o from subtree ConfigNC() where
125. o!objectGUID = dsaObj!parent
126. msgOut^.rItems[i].pName := serverObj.dn
127. msgOut^.rItems[i].status := DS\_NAME\_NO\_ERROR
128. i := i + 1
129. endfor
130. msgOut^.cItems := i
131. else if msgIn.formatOffered = DS\_LIST\_INFO\_FOR\_SERVER then
132. /\* Returns the DSA object, the dnsHostName and the serverReference
133. \* for the server specified by msgIn.rpNames[0]. \*/
134. serverObj := GetDSNameFromDN(msgIn.rpNames[0])
135. dsaObj := select one o from subtree msgIn.rpNames[0]
136. where o!objectCategory = GetDefaultObjectCategory(nTDSDSA)
137. if dsaObj ≠ null then
138. /\* Ok, looks like a valid server object. \*/
139. msgOut^.rItems[0].pName := dsaObj.dn
140. msgOut^.rItems[0].status := DS\_NAME\_NO\_ERROR
141. msgOut^.rItems[1].pName := serverObj!dnsHostName
142. msgOut^.rItems[1].status := DS\_NAME\_NO\_ERROR
143. msgOut^.rItems[2].pName := serverObj!serverReference
144. msgOut^.rItems[2].status := DS\_NAME\_NO\_ERROR
145. msgOut^.cItems := 3
146. endif
147. else if msgIn.formatOffered = DS\_LIST\_GLOBAL\_CATALOG\_SERVERS then
148. /\* Returns the list of GC servers, including the info which site
149. \* each GC belongs to. \*/
150. rt := select all o from subtree ConfigNC()
151. where O!objectCategory = GetDefaultObjectCategory(nTDSDSA)
152. and NTDSDSA\_OPT\_IS\_GC in o!options and o!invocationId ≠ null
153. i := 0
154. foreach dsaObj in rt
155. /\* server object is the parent of the DSA object. \*/
156. serverObj := select one o from subtree ConfigNC() where
157. o!objectGUID = dsaObj!parent
158. /\* Site object is the parent of the server object. \*/
159. siteObj := select one o from subtree ConfigNC() where
160. o!objectGUID = serverObj!parent
161. msgOut^.rItems[i].pDomain := serverObj!dnsHostName
162. msgOut^.rItems[i].pName := leftmost RDN of siteObj.dn
163. msgOut^.rItems[i].status := DS\_NAME\_NO\_ERROR
164. i := i+1
165. endfor
166. msgOut.cItems := i
167. else if msgIn.formatOffered = DS\_MAP\_SCHEMA\_GUID then
168. for i := 0 to msgIn.cNames - 1
169. /\* Map a guid contained in msgIn.rpNames[i] to attribute or class
170. \* or propertySet.\*/
171. /\* Assume no match by default. \*/
172. msgOut^.rItems[i].status := DS\_NAME\_ERROR\_SCHEMA\_GUID\_NOT\_FOUND
174. /\* Validate the string guid contained in msgIn.rpNames[i] \*/
175. guid := GuidFromString(true, msgIn.rpNames[i])
176. if guid ≠ null then
178. /\* First, try to find a matching attribute. \*/
179. attr := select one o from subtree SchemaNC()
180. where attributeSchema in o!objectClass and
181. o!schemaIdGuid = msgIn.rpNames[i]
182. if attr ≠ null
183. /\* Found a matching attribute object. \*/
184. msgOut^.rItems[i].pName := attr!lDAPDisplayName
185. msgOut^.rItems[i].status := DS\_NAME\_ERROR\_SCHEMA\_GUID\_ATTR
186. else
187. /\* Next, try to find a matching class. \*/
188. class := select one o from subtree SchemaNC()
189. where classSchema in o!objectClass
190. o!schemaIdGuid = msgIn.rpNames[i]
191. if class ≠ null
192. /\* Found a matching class object. \*/
193. msgOut^.rItems[i].pName := class!lDAPDisplayName
194. msgOut^.rItems[i].status := DS\_NAME\_ERROR\_SCHEMA\_GUID\_CLASS
195. else
196. /\* Finally, try to find a matching extendedRight object. \*/
197. er := select one o from
198. subtree DescendantObject(ConfigNC(),
199. "CN=Extended-Rights,")
200. where extendedRight in o!objectClass and
201. o!rightsGuid = msgIn.rpNames[i]
202. if er ≠ null
203. /\* Found a matching extendedRight object \*/
204. if RIGHT\_DS\_READ\_PROPERTY in er!validAccesses or
205. RIGHT\_DS\_WRITE\_PROPERTY in er!validAccesses then
206. msgOut^.rItems[i].pName := er!displayName
207. msgOut^.rItems[i].status :=
208. DS\_NAME\_ERROR\_SCHEMA\_GUID\_ATTR\_SET
209. else if RIGHT\_DS\_CONTROL\_ACCESS in er!validAccesses or
210. RIGHT\_DS\_WRITE\_PROPERTY\_EXTENDED in er!validAccesses
211. then
212. msgOut^.rItems[i].pName := er!displayName
213. msgOut^.rItems[i].status :=
214. DS\_NAME\_ERROR\_SCHEMA\_GUID\_CONTROL\_RIGHT
215. endif
216. endif
217. endif
218. endif
219. endif
220. endfor
221. msgOut^.cItems := msgIn.cNames
222. endif
223. return ERROR\_SUCCESS

##### LookupName

1. procedure LookupName(
2. flags: DWORD,
3. formatOffered: DWORD,
4. formatDesired: DWORD,
5. name: unicodestring): DS\_NAME\_RESULT\_ITEMW

*Informative summary of behavior*: The LookupName procedure performs the lookup of a single *name* in a given input format and produces the output *name* in the given output format.

1. rt: sequence of DSName
2. obj: DSName
3. fSidHistory: boolean
4. result: DS\_NAME\_RESULT\_ITEMW
5. names: sequence of unicodestring
6. domainName: unicodestring
7. fCanonicalEx: boolean
8. referredDomain: unicodestring
9. if formatOffered = DS\_UNKNOWN\_NAME then
10. return LookupUnknownName(flags, name, formatDesired)
11. endif
12. domainName := null
13. if formatOffered = DS\_FQDN\_1779\_NAME then
14. rt := LookupAttr(flags, distinguishedName, name)
15. domainName := DomainDNSNameFromDomain(RetrieveDCSuffixFromDn(name))
16. else if formatOffered = DS\_NT4\_ACCOUNT\_NAME then
17. rt := LookupAttr(flags, sAMAccountName,
18. UserNameFromNT4AccountName(name))
19. domainName := DomainNameFromNT4AccountName(name)
20. else if formatOffered = DS\_USER\_PRINCIPAL\_NAME then
21. rt := LookupUPNAndAltSecID(flags, false, name)
22. domainName := DomainNameFromUPN(name)
23. else if formatOffered = DS\_CANONICAL\_NAME then
24. rt := LookupCanonicalName(name)
25. domainName := DomainNameFromCanonicalName(name)
26. else if formatOffered = DS\_UNIQUE\_ID\_NAME then
27. rt := select all o from all where o!objectGuid = GuidFromString(true, name)
28. else if formatOffered = DS\_DISPLAY\_NAME then
29. rt := LookupAttr(flags, displayName, name)
30. else if formatOffered = DS\_SERVICE\_PRINCIPAL\_NAME then
31. rt := LookupSPN(flags, name)
32. domainName := GetServiceNameFromSPN(name)
33. else if formatOffered in {DS\_SID\_OR\_SID\_HISTORY\_NAME,
34. DS\_STRING\_SID\_NAME} then
35. rt := LookupSID(flags, SidFromStringSid(name))
36. domainName := DomainNameFromSid(DomainSidFromSid(SidFromStringSid(name)))
37. else if formatOffered = DS\_CANONICAL\_NAME\_EX then
38. rt := LookupCanonicalName(CanonicalNameFromCanonicalNameEx(name))
39. domainName := DomainNameFromCanonicalName(name)
40. else if formatOffered in {DS\_NT4\_ACCOUNT\_NAME\_SANS\_DOMAIN,
41. DS\_NT4\_ACCOUNT\_NAME\_SANS\_DOMAIN\_EX} then
42. rt := LookupAttr(flags, sAMAccountName, name)
43. else if formatOffered = DS\_ALT\_SECURITY\_IDENTITIES\_NAME then
44. rt := LookupAttr(flags, altSecurityIdentities, name)
45. else if formatOffered = DS\_USER\_PRINCIPAL\_NAME\_AND\_ALTSECID then
46. rt := LookupUPNAndAltSecID(flags, true, name)
47. domainName := DomainNameFromUPN(name)
48. else
49. rt := null
50. endif
51. result.pName^ := null
52. result.pDomain^ := null
53. result.status := DS\_NAME\_NO\_ERROR
54. if rt = null and domainName ≠ null then
55. result.status := DS\_NAME\_ERROR\_DOMAIN\_ONLY
56. if formatOffered in {DS\_NT4\_ACCOUNT\_NAME, DS\_USER\_PRINCIPAL\_NAME,
57. DS\_SERVICE\_PRINCIPAL\_NAME, DS\_SID\_OR\_SID\_HISTORY\_NAME,
58. DS\_STRING\_SID\_NAME,DS\_USER\_PRINCIPAL\_NAME\_AND\_ALTSECID} then
59. if IsDomainNameInTrustedForest(domainName, referredDomain) then
60. result.pDomain^ := referredDomain
61. if DS\_NAME\_FLAG\_TRUST\_REFERRAL in flags then
62. result.status := DS\_NAME\_ERROR\_TRUST\_REFERRAL
63. else
64. result.status := DS\_NAME\_ERROR\_DOMAIN\_ONLY
65. endif
66. endif
67. endif
68. return result
69. endif
70. if rt = null then
71. /\* No match. \*/
72. result.status := DS\_NAME\_ERROR\_NOT\_FOUND
73. return result
74. endif
75. if rt.length > 1 then
76. /\* Found more than one matching object. \*/
77. result.status := DS\_NAME\_ERROR\_NOT\_UNIQUE
78. return result
79. endif
80. obj := rt[0]
81. if formatOffered = DS\_NT4\_ACCOUNT\_NAME\_SANS\_DOMAIN\_EX then
82. /\* Check that the account is valid. \*/
83. if obj!userAccountControl ∩ {ADS\_UF\_ACCOUNTDISABLE,
84. ADS\_UF\_TEMP\_DUPLICATE\_ACCOUNT} ≠ {} then
85. result.status := DS\_NAME\_ERROR\_NOT\_FOUND
86. return result
87. endif
88. endif
89. if formatOffered = DS\_STRING\_SID\_NAME then
90. /\* The type of the object needs to be specified in result.status. \*/
91. /\* Check if the value came from sIDHistory or objectSid. \*/
92. fSidHistory := SidFromStringSid(name) in obj!sidHistory
93. if obj!sAMAccountType in {SAM\_USER\_OBJECT, SAM\_MACHINE\_ACCOUNT,
94. SAM\_TRUST\_ACCOUNT} then
95. if fSidHistory then
96. result.status := DS\_NAME\_ERROR\_IS\_SID\_HISTORY\_USER
97. else
98. result.status := DS\_NAME\_ERROR\_IS\_SID\_USER
99. endif
100. else if obj!sAMAccountType in {SAM\_NON\_SECURITY\_GROUP\_OBJECT,
101. SAM\_GROUP\_OBJECT} then
102. if fSidHistory then
103. result.status := DS\_NAME\_ERROR\_IS\_SID\_HISTORY\_GROUP
104. else
105. result.status := DS\_NAME\_ERROR\_IS\_SID\_GROUP
106. endif
107. else if obj!sAMAccountType in {SAM\_NON\_SECURITY\_ALIAS\_OBJECT,
108. SAM\_ALIAS\_OBJECT} then
109. if fSidHistory then
110. result.status := DS\_NAME\_ERROR\_IS\_SID\_HISTORY\_ALIAS
111. else
112. result.status := DS\_NAME\_ERROR\_IS\_SID\_ALIAS
113. endif
114. else
115. if fSidHistory then
116. result.status := DS\_NAME\_ERROR\_IS\_SID\_HISTORY\_UNKNOWN
117. else
118. result.status := DS\_NAME\_ERROR\_IS\_SID\_UNKNOWN
119. endif
120. endif
121. endif
122. /\* Found exactly one object. Construct the output name in the
123. \* desired format. \*/
124. names := ConstructOutput(obj, formatDesired)
125. if formatDesired not in { DS\_FQDN\_1779\_NAME, DS\_NT4\_ACCOUNT\_NAME, DS\_DISPLAY\_NAME, DS\_UNIQUE\_ID\_NAME, DS\_CANONICAL\_NAME, DS\_CANONICAL\_NAME\_EX, DS\_USER\_PRINCIPAL\_NAME, DS\_SERVICE\_PRINCIPAL\_NAME, DS\_STRING\_SID\_NAME, DS\_USER\_PRINCIPAL\_NAME\_FOR\_LOGON } then
126. result.status := DS\_NAME\_ERROR\_RESOLVING
127. return result
128. end if
129. if names = null and
130. foreignSecurityPrincipal in obj!objectClass and
131. obj!SID ≠ null and
132. DS\_NAME\_FLAG\_PRIVATE\_RESOLVE\_FPOS in flags and
133. formatDesired in { DS\_NT4\_ACCOUNT\_NAME, DS\_DISPLAY\_NAME,
134. DS\_CANONICAL\_NAME, DS\_CANONICAL\_NAME\_EX, DS\_USER\_PRINCIPAL\_NAME,
135. DS\_USER\_PRINCIPAL\_NAME\_FOR\_LOGON, DS\_SERVICE\_PRINCIPAL\_NAME} then
136. /\* Found a foreign security principal for which the desired name is not
137. \* included. Use the LSAT protocol to lookup the name. Note: For any
138. \* desired format, it can only return either DS\_CANONICAL\_NAME or
139. \* DS\_CANONICAL\_NAME\_EX. \*/
140. if (formatDesired=DS\_CANONICAL\_NAME\_EX) then
141. fCanonicalEx := true
142. else
143. fCanonicalEx := false
144. endif
145. result := LookupFPO(fCanonicalEx, obj, result)
146. return result
147. endif
148. if names = null then
149. /\* Could not construct the required name format. \*/
150. result.status := DS\_NAME\_ERROR\_NO\_MAPPING
151. return result
152. endif
153. if names.length > 1 then
154. /\* Too many output names. \*/
155. result.status := DS\_NAME\_ERROR\_NOT\_UNIQUE
156. return result
157. endif
158. result.pName^ := names[0]
159. result.pDomain^ := DomainDNSNameFromDomain(GetObjectNC(obj))
160. result.status = DS\_NAME\_NO\_ERROR
161. return result

##### LookupAttr

1. procedure LookupAttr(
2. flags: DWORD,
3. att: ATTRTYP,
4. attrValue: unicodestring): set of DSName

*Informative summary of behavior*: The LookupAttr procedure is a helper function that looks up an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) based on an attributeName=attributeValue criterion. It returns the set of objects that match the criterion.

1. rt: set of DSName
2. if DS\_NAME\_FLAG\_GCVERIFY in flags or IsGC() then
3. rt := select all O from all
4. where attrValue in GetAttrVals(O, att, false)
5. else
6. rt := select all O from subtree DefaultNC()
7. where attrValue in GetAttrVals(O, att, false)
8. endif
9. return rt

##### LookupCanonicalName

1. procedure LookupCanonicalName(name: unicodestring): DSName

*Informative summary of behavior*: The LookupCanonicalName procedure is a helper function that looks up an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) based on its [**canonical name**](#gt_79ab9d86-0d30-41c3-b7da-153ad41bdfd8) by walking down the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) from the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root and looking up objects by *name*.

1. curObj: DSName
2. label: unicodestring
3. ParseCanonicalName(name, label, name)
4. curObj := DomainFromDomainDNSName(label)
5. while name ≠ null and curObj ≠ null
6. ParseCanonicalName(name, label, name)
7. curObj := select one O from children curObj where O!name=label
8. if curObj = null then
9. return null
10. endif
11. endwhile
12. return curObj

##### GetCanonicalName

1. procedure GetCanonicalName(
2. obj: DSName, extended: boolean): unicodestring

*Informative summary of behavior*: The GetCanonicalName function constructs the [**canonical name**](#gt_79ab9d86-0d30-41c3-b7da-153ad41bdfd8) of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) by walking up its ancestors to the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root.

1. result: unicodestring
2. if obj = GetDomainNC(obj) then
3. return DomainDNSNameFromDomain(obj)
4. endif
5. /\* Recurse into parent, obtain non-extended canonical name. \*/
6. result := GetCanonicalName(obj!parent, false)
7. if extended = true then
8. result := result + "\n"
9. else
10. result := result + "/"
11. endif
12. result := result + obj!name
13. return result

##### LookupSPN

1. procedure LookupSPN(flags: DWORD, name: unicodestring): set of DSName

*Informative summary of behavior*: LookupSPN is a helper function that implements the [**service principal name (SPN)**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) lookup algorithm.

1. rt: set of DSName
2. obj: DSName
3. dcGuid: GUID
4. spnMappings: set of unicodestring
5. mappedSpn: unicodestring
6. /\* First, try to look up the SPN directly. \*/
7. rt := LookupAttr(flags, servicePrincipalName, name)
8. if rt ≠ null then
9. return rt
10. endif
11. /\* Obtain SPN mappings value. \*/
12. obj := DescendantObject(ConfigNC(),
13. "CN=Directory Service,CN=Windows NT,CN=Services,")
14. spnMappings := obj!sPNMappings
15. if spnMappings ≠ null
16. mappedSpn := MapSPN(name, spnMappings)
17. if mappedSpn ≠ null then
18. /\* try to lookup a mapped SPN \*/
19. rt := LookupAttr(flags, servicePrincipalName, mappedSpn)
20. if rt ≠ null then
21. return rt
22. endif
23. endif
24. endif
25. /\* Try to find replication SPN, which might not be present in our
26. \* NC replicas yet. \*/
27. if GetServiceClassFromSPN(name) = DRS\_SPN\_CLASS and
28. GetServiceNameFromSPN(name) =
29. DomainNameFromDN(DefaultNC()!distinguishedName) then
30. /\* Yes, it looks like a replication SPN. Try to find DC by guid. \*/
31. dcGuid := GuidFromString(false, GetInstanceNameFromSPN(name))
32. if dcGuid ≠ null then
33. /\* Find DSA object with this objectGUID value. \*/
34. obj := select one o from subtree ConfigNC()
35. where o!objectGUID = dcGuid
36. if obj ≠ null then
37. /\* Get the server object. \*/
38. obj := obj!parent
39. if obj ≠ null then
40. /\* server!serverReference points to the DC's computer
41. \* object.\*/
42. rt := {obj!serverReference}
43. endif
44. endif
45. endif
46. endif
47. return rt

##### LookupSID

1. procedure LookupSID(flags: DWORD, sid: SID): set of DSName

*Informative summary of behavior*: The LookupSID procedure is a helper function that implements the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) lookup algorithm.

1. rt1, rt2: set of DSName
2. rt1 := LookupAttr(flags, objectSid, sid)
3. rt2 := LookupAttr(flags, sIDHistory, sid)
5. return rt1 + rt2

##### LookupUnknownName

1. procedure LookupUnknownName(
2. flags: DWORD,
3. name: unicodestring,
4. formatDesired: DWORD): DS\_NAME\_RESULT\_ITEMW

*Informative summary of behavior*: The server uses LookupUnknownName to look up names of format DS\_UNKNOWN\_NAME. LookupUnknownName looks up the *name* by trying formats in the specific order listed in the foreach statement shown below until a lookup succeeds and produces the output *name* in the given output format.

1. result: DS\_NAME\_RESULT\_ITEMW
2. format: DWORD
3. /\* Attempt to resolve in the following formats in this specific
4. \* order. \*/
5. foreach format in {DS\_FQDN\_1779\_NAME, DS\_USER\_PRINCIPAL\_NAME,
6. DS\_NT4\_ACCOUNT\_NAME, DS\_CANONICAL\_NAME,
7. DS\_UNIQUE\_ID\_NAME, DS\_DISPLAY\_NAME,
8. DS\_SERVICE\_PRINCIPAL\_NAME,
9. DS\_SID\_OR\_SID\_HISTORY\_NAME,
10. DS\_CANONICAL\_NAME\_EX}
11. result := LookupName(flags, format, formatDesired, name)
12. if result.status ≠ DS\_NAME\_ERROR\_NOT\_FOUND then
13. return result
14. endif
15. endfor
16. return result

##### LookupUPNAndAltSecID

1. procedure LookupUPNAndAltSecID(
2. flags: DWORD,
3. IncludingAltSecID: boolean,
4. name: unicodestring): set of DSName

*Informative summary of behavior*: Returns [DSName](#Section_a0d5477a522946b9890a54b924d487d1)s of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca), with the given value as a value of userPrincipalName, altSecurityIdentities, or sAMAccountName.

1. rt, rt1, rt2: set of DSName
2. /\* Try lookup by userPrincipalName and altSecurityIdentities
3. \* or by only userPrincipalName depending on what is
4. \* requested \*/
5. if IncludingAltSecID then
6. rt1 := LookupAttr(flags, userPrincipalName, name)
7. rt2 := LookupAttr(flags, altSecurityIdentities, name)
8. rt := rt1 + rt2
9. else
10. rt := LookupAttr(flags, userPrincipalName, name)
11. endif
12. if rt ≠ null then
13. return rt
14. endif
15. /\* Finally, attempt to parse the name as simpleName@domain and
16. \* search for
17. \* sAMAccountName=simpleName. \*/
18. name := UserNameFromUPN(name)
19. if name ≠ null then
20. rt := LookupAttr(flags, sAMAccountName, name)
21. endif
22. return rt

##### LookupFPO

1. procedure LookupFPO(
2. fCanonicalEx: boolean,
3. obj: DSName,
4. result: DS\_NAME\_RESULT\_ITEMW
5. ): DS\_NAME\_RESULT\_ITEMW

*Informative summary of behavior*: LookupFPO is a helper function that attempts to resolve the domain and account name of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca), with an appropriate status value.

1. pReferencedDomain: PLSAPR\_REFERENCED\_DOMAIN\_LIST
2. TranslatedName: LSAPR\_TRANSLATED\_NAMES\_EX
3. NtStatus: NTSTATUS
4. pReferencedDomain := null
5. TranslatedName := null
6. NtStatus := TranslateFPOToName(obj, ADR(pReferencedDomain),
7. ADR(TranslatedName))
8. if (NtStatus = 0x0 and pReferencedDomain ≠ null and TranslatedName ≠ null)
9. then
10. result.pDomain^ :=
11. pReferencedDomains^.Domains[TranslatedName.DomainIndex].Name
12. if fCanonicalNameEx then
13. result.pName^ := result.pDomain^ + {"\n"} + TranslatedName.Name
14. else
15. result.pName^ := result.pDomain^ + {"\"} + TranslatedName.Name
16. endif
17. result.status := DS\_NAME\_ERROR\_IS\_FPO
18. return result
19. else
20. result.status := DS\_NAME\_ERROR\_RESOLVING
21. endif
22. /\* leave result as-is. \*/
23. return result

##### MapSPN

1. procedure MapSPN(spn: unicodestring,
2. spnMappings: set of unicodestring):
3. unicodestring

The MapSPN procedure performs an [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) mapping operation on *spn* according to the map specified in *spnMappings*, and returns the mapped version of *spn*. The mapping operation is used to change the [**service class**](#gt_c1386afd-9d42-47c7-a7ea-d01912a17647) of the SPN. An SPN service class is the first part of an SPN; for example, "ldap" is the service class of the SPN "ldap/fabrikam.com".

Each value of *spnMappings* consists of an alias, followed by an equals sign (=), followed by a comma-separated list of one or more SPN service classes. Thus, each value must be in the following format:

alias=serviceClass1,serviceClass2,serviceClass3,...,serviceClassN

If the service class portion of *spn* corresponds to one of the serviceClassX values in value *v* of *spnMappings*, then the return value of this procedure is the SPN value this is constructed from *spn* by substituting the alias value from *v* as the service class of *spn*. If no mapping is found (that is, if there is no such *v*), or if *spn* is not an SPN, then null is returned.

For example, suppose that *spnMappings* is the following set:

{"ldap=ldap,otherldap", "host=alerter,appmgmt,cisvc"}

If spn is "alerter/fabrikam.com", then the procedure returns "host/fabrikam.com".

##### ParseCanonicalName

1. procedure ParseCanonicalName(
2. name: unicodestring,
3. var firstPart: unicodestring,
4. var remainder: unicodestring)

The ParseCanonicalName procedure parses the first label from the [**canonical name**](#gt_79ab9d86-0d30-41c3-b7da-153ad41bdfd8) string *name* and returns the first label in *firstPart* and the *remainder* of the string in *remainder*. For example, *name* = "container1/container2/leaf" is parsed as *firstPart*:= "container1" and *remainder*:= "container2/leaf". As another example, *name* = "example.fabrikam.com/container/username" is parsed as *firstPart*:= "example.fabrikam.com" and *remainder*:= "container/username". If *name* does not contain a slash character, then it is parsed as *firstPart*:= *name* and *remainder*:= null.

##### RetrieveDCSuffixFromDn

1. procedure RetrieveDCSuffixFromDn(dn: unicodestring): unicodestring

The RetrieveDCSuffixFromDn procedure parses the [**distinguished name (DN)**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) syntactically and returns the suffix that consists entirely of the DN components whose [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) type is "DC". For example, given "CN=Administrator,CN=Users,DC=fabrikam,DC=com", this procedure would return "DC=fabrikam,DC=com".

##### UserNameFromUPN

1. procedure UserNameFromUPN(upn: unicodestring): unicodestring

Parses and returns the user name from a UPN-formatted string *upn*. The user name is the component before the '@'. For example, when the input is "username@example.fabrikam.com", then "username" is returned. If the input is not in UPN format, then null is returned.

##### TranslateFPOToName

1. procedure TranslateFPOToName(
2. obj: DSName,
3. ppReferencedDomains: PLSAPR\_REFERENCED\_DOMAIN\_LIST\*,
4. pTranslatedNames: PLSAPR\_TRANSLATED\_NAMES\_EX
5. ): NTSTATUS

*Informative summary of behavior:* The TranslateFPOToName procedure performs an LsarLookupSids2 call ([[MS-LSAT]](%5bMS-LSAT%5d.pdf#Section_1ba21e6fd8a9462c91534375f2020894) section 3.1.4.10) to translate *obj* to its Windows NT 4.0 account name and [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

1. hlsaPolicy: LSAPR\_HANDLE
2. mappedCount: unsigned long
3. systemName: unicodeString
4. objectAttributes: LSAPR\_OBJECT\_ATTRIBUTES
5. desiredAccess: DWORD
6. sidEnumBuffer: LSAPR\_SID\_ENUM\_BUFFER
7. sidInfo: LSAPR\_SID\_INFORMATION
8. NtStatus: NTSTATUS
9. sidEnumBuffer.Entries := 1
10. sidInfo.Sid := obj!Sid
11. sidEnumBuffer.SidInfo := ADR(sidInfo)
12. systemName := ""
13. objectAttributes.Length := 0
14. objectAttributes.RootDirectory := null
15. objectAttributes.ObjectName := null
16. objectAttributes.attributes := 0
17. objectAttributes.SecurityDescriptor := null
18. objectAttributes.SecurityQualityOfService := null
19. desiredAccess := 0x00000800
20. NtStatus := LsarOpenPolicy2(systemName, ADR(objectAttributes),
21. desiredAccess, ADR(hlsaPolicy))
22. if 0x0 = NtStatus then
23. NtStatus := LsarLookupSids2(hlsaPolicy, ADR(sidEnumBuffer),
24. ppReferencedDomains, pTranslatedNames,
25. 0x1, ADR(mappedCount), 0x0, 0x2)
26. endif
27. If hlsaPolicy ≠ null
28. LsarClose(ADR(hlsaPolicy))
29. return NtStatus

##### ConstructOutput

1. procedure ConstructOutput(
2. obj: DSName,
3. formatDesired: DWORD): set of unicodestring

*Informative summary of behavior*: ConstructOutput is a helper function that constructs the name of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the required output format. Note that the returned set of values might be empty or might contain more than one value. These situations are handled by the caller function, [LookupName (section 4.1.4.2.10)](#Section_a6a070e9e2434bcb8bc7b2774c18a729).

1. if formatDesired = DS\_FQDN\_1779\_NAME then
2. return {obj!distinguishedName}
3. else if formatDesired = DS\_NT4\_ACCOUNT\_NAME then
4. if obj!sAMAccountName ≠ null then
5. return {DomainNetBIOSNameFromDomain(GetObjectNC(obj)) + "\" +
6. obj!sAMAccountName}
7. else if IsDomainOnly(obj) then
8. return {DomainNetBIOSNameFromDomain(GetObjectNC(obj)) + "\"}
9. else if formatDesired = DS\_USER\_PRINCIPAL\_NAME then
10. return {obj!userPrincipalName}
11. else if formatDesired = DS\_CANONICAL\_NAME then
12. return {GetCanonicalName(obj, false)}
13. else if formatDesired = DS\_UNIQUE\_ID\_NAME then
14. return {GuidToString(obj!objectGUID)}
15. else if formatDesired = DS\_DISPLAY\_NAME then
16. return {obj!displayName}
17. else if formatDesired = DS\_SERVICE\_PRINCIPAL\_NAME then
18. return obj!servicePrincipalName
19. else if formatDesired = DS\_CANONICAL\_NAME\_EX then
20. return {GetCanonicalName(obj, true)}
21. else if formatDesired = DS\_STRING\_SID\_NAME then
22. return {StringSidFromSid(obj!objectSid)}
23. else if formatDesired = DS\_USER\_PRINCIPAL\_NAME\_FOR\_LOGON then
24. /\* If UPN is set, then return it. \*/
25. if obj!userPrincipalName ≠ null then
26. return {obj!userPrincipalName}
27. endif
28. return {obj!sAMAccountName + "@" +
29. DomainDNSNameFromDomain(GetObjectNC(obj))}
30. endif
31. /\* Otherwise, unknown format. \*/
32. return null

##### IsDomainOnly

*Informative summary of behavior*: The function determines whether the given DSName is one of the [**domain NCs**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) present in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62).

1. procedure IsDomainOnly(obj: DSName): boolean
2. cr: DSName
3. /\* Confirm that obj is a domainNC in the forest of the server. \*/
4. cr := select one domainNC from subtree ConfigNC() where
5. (crossRef in domainNC!objectClass and
6. domainNC!nCName = obj!distinguishedName)
7. if cr = null then
8. return FALSE
9. else
10. return TRUE
11. endif

#### Server Behavior of the IDL\_DRSCrackNames Method

*Informative summary of behavior*: The IDL\_DRSCrackNames method is a generic method that is used to look up information in the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9). The most common usage is looking up [**directory object**](#gt_407dbc2c-3140-4e31-9085-0087e2d3bab2) names that are provided in one format (for example, [**SPNs**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4)) and returning them in a different format (for example, [**DNs**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b)). One special mode occurs when the input format is not specified, in which case the server tries to "guess" the format of the name by following some heuristics. The method can also be used to look up generic information in the directory, such as the list of [**sites**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba) or the list of servers in a specific site.

1. ULONG
2. IDL\_DRSCrackNames(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_CRACKREQ \*pmsgIn,
6. [out, ref] DWORD \*pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_CRACKREPLY \*pmsgOut)
9. msgIn: DRS\_MSG\_CRACKREQ\_V1
10. msgOut: DS\_NAME\_RESULTW
11. ULONG result
12. ValidateDRSInput(hDrs, 12)
13. pdwOutVersion^ := 1
14. pmsgOut^.V1.pResult^.cItems := 0
15. pmsgOut^.V1.pResult^.rItems := null
16. if dwInVersion ≠ 1 then
17. return ERROR\_INVALID\_PARAMETER
18. endif
19. msgIn := pmsgIn^.V1
20. if DS\_NAME\_FLAG\_GCVERIFY in msgIn.dwFlags and
21. not IsGC() then
22. return ERROR\_DS\_GCVERIFY\_ERROR
23. endif
24. /\* Enable FPO resolution for non-DC callers. \*/
25. if ClientUUID(hDrs) = NTDSAPI\_CLIENT\_GUID then
26. msgIn.dwFlags := msgIn.dwFlags + {DS\_NAME\_FLAG\_PRIVATE\_RESOLVE\_FPOS}
27. endif
28. result = CrackNames(pmsgIn^.V1, ADR(msgOut))
29. if(result = ERROR\_SUCCESS) then
30. pmsgOut^.V1.pResult := ADR(msgOut)
31. endif
32. return result

#### Examples of the IDL\_DRSCrackNames Method

When user "Kim Akers" logs on to the computer MS1.Contoso.com using her Windows NT 4.0 account name "CONTOSO\kimakers", the [**domain controller**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) needs to obtain a [**fully qualified domain name (FQDN) (1)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) that corresponds to the Windows NT 4.0 account name. The domain controller DC1 calls IDL\_DRSCrackNames to translate the Windows NT 4.0 account name to an FQDN (1).

##### Initial State

Querying the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) with name KimAkers in the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) DC=CONTOSO, DC=COM on DC1:

* ldap\_search\_s("CN=Kim Akers,CN=Users,DC=contoso,DC=com", *baseObject*, "(objectClass=user)", [*objectClass, distinguishedName, sAMAccountName*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=Kim Akers,CN=Users,DC=contoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> distinguishedName: CN=Kim Akers,CN=Users,DC=contoso,DC=com;
  + 1> sAMAccountName: KimAkers

Querying the [**crossRef object**](#gt_353fac65-0774-4ba8-8081-eb4c963f94e7) for the domain NC CONTOSO.COM on DC1 by performing the following [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) search:

* ldap\_search\_s("CN=CONTOSO,CN=Partitions,CN=Configuration,DC=contoso,DC=com", baseObject, "(objectClass=crossRef)", [objectClass, nCName, dnsRoot, nETBIOSName])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=CONTOSO,CN=Partitions,CN=Configuration,DC=contoso,DC=com
  + 2> objectClass: top; crossRef;
  + 1> nCName: DC=contoso,DC=com;
  + 1> dnsRoot: contoso.com;
  + 1> nETBIOSName: CONTOSO;

##### Client Request

DC1 invokes the IDL\_DRSCrackNames method against itself with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 1
* *pmsgIn* = DRS\_MSG\_CRACKREQ\_V1
  + CodePage = 0x4e4
  + LocaleId = US-EN
  + dwFlags = 0
  + formatOffered = DS\_NT4\_ACCOUNT\_NAME
  + formatDesired = DS\_FQDN\_1779\_NAME
  + cNames: 1
  + rpNames: "CONTOSO\kimakers"

##### Server Response

Returns code of 0 and the following values:

* *pdwMessageOut* = 1

*pmsgOut* = DRS\_MSG\_CRACKREPLY\_V1

* + pResult: DS\_NAME\_RESULTW
  + cNames: 1
  + rItems: DS\_NAME\_RESULT\_ITEMW
  + pDomain: "contoso.com"
  + pName: "CN=Kim Akers,CN=Users,DC=contoso,DC=com"
  + status: DS\_NAME\_NO\_ERROR

##### Final State

The final state is the same as the initial state; there is no change.

### IDL\_DRSDomainControllerInfo (Opnum 16)

The IDL\_DRSDomainControllerInfo method retrieves information about [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) in a given [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

1. ULONG IDL\_DRSDomainControllerInfo(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_DCINFOREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_DCINFOREPLY\* pmsgOut
9. );

**hDrs:** [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** Version of the request message.

**pmsgIn:** Pointer to the request message.

**pdwOutVersion:** Pointer to the version of the response message.

**pmsgOut:** Pointer to the response message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_DCINFOREQ

The DRS\_MSG\_DCINFOREQ union defines the request messages sent to the [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_DCINFOREQ\_V1 V1;
6. } DRS\_MSG\_DCINFOREQ,
7. \*PDRS\_MSG\_DCINFOREQ;

**V1:**  Version 1 request.

##### DRS\_MSG\_DCINFOREQ\_V1

The DRS\_MSG\_DCINFOREQ\_V1 structure defines the request message sent to the [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) method.

1. typedef struct {
2. [string] WCHAR\* Domain;
3. DWORD InfoLevel;
4. } DRS\_MSG\_DCINFOREQ\_V1;

**Domain:**  The [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) for which the client requests information. The domain can be an [**FQDN (1)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) or a [**NetBIOS domain name**](#gt_f7f8efcc-c6d5-40f0-9605-c9d99c5a0b92).

**InfoLevel:**  The response version requested by the client: 1, 2, 3, or 0xFFFFFFFF. The responses at InfoLevel 1, 2, and 3 all contain information about [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) in the given domain. The information at InfoLevel 1 is a subset of the information at InfoLevel 2, which is a subset of the information at InfoLevel 3. InfoLevel 3 includes information about the [**RODCs**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870) in the given domain. InfoLevel 0xFFFFFFFF server returns information about the active [**LDAP connections**](#gt_198f4791-cea3-465d-89e2-262991624e08).

##### DRS\_MSG\_DCINFOREPLY

The DRS\_MSG\_DCINFOREPLY union defines the response messages received from the [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_DCINFOREPLY\_V1 V1;
6. [case(2)]
7. DRS\_MSG\_DCINFOREPLY\_V2 V2;
8. [case(3)]
9. DRS\_MSG\_DCINFOREPLY\_V3 V3;
10. [case(0xFFFFFFFF)]
11. DRS\_MSG\_DCINFOREPLY\_VFFFFFFFF VFFFFFFFF;
12. } DRS\_MSG\_DCINFOREPLY;

**V1:**  Version 1 response.

**V2:**  Version 2 response.

**V3:**  Version 3 response.

**VFFFFFFFF:**  Version 0xFFFFFFFF response.

##### DRS\_MSG\_DCINFOREPLY\_V1

The DRS\_MSG\_DCINFOREPLY\_V1 structure defines the response message received from the [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) method, when the client has requested InfoLevel = 1.

1. typedef struct {
2. [range(0,10000)] DWORD cItems;
3. [size\_is(cItems)] DS\_DOMAIN\_CONTROLLER\_INFO\_1W\* rItems;
4. } DRS\_MSG\_DCINFOREPLY\_V1;

**cItems:**  Count of items in the **rItems** array.

**rItems:**  [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) information.

##### DRS\_MSG\_DCINFOREPLY\_V2

The DRS\_MSG\_DCINFOREPLY\_V2 structure defines the response message received from the [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) method, when the client has requested InfoLevel = 2.

1. typedef struct {
2. [range(0,10000)] DWORD cItems;
3. [size\_is(cItems)] DS\_DOMAIN\_CONTROLLER\_INFO\_2W\* rItems;
4. } DRS\_MSG\_DCINFOREPLY\_V2;

**cItems:**  Count of items in the **rItems** array.

**rItems:**  [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) information.

##### DRS\_MSG\_DCINFOREPLY\_V3

The DRS\_MSG\_DCINFOREPLY\_V3 structure defines the response message received from the [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) method when the client has requested InfoLevel = 3.

1. typedef struct {
2. [range(0,10000)] DWORD cItems;
3. [size\_is(cItems)] DS\_DOMAIN\_CONTROLLER\_INFO\_3W\* rItems;
4. } DRS\_MSG\_DCINFOREPLY\_V3;

**cItems:**  Count of items in the **rItems** array.

**rItems:**  [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) information.

##### DRS\_MSG\_DCINFOREPLY\_VFFFFFFFF

The DRS\_MSG\_DCINFOREPLY\_VFFFFFFFF structure defines the response message received from the [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) method, when the client has requested InfoLevel = 0xFFFFFFFF.

1. typedef struct {
2. [range(0,10000)] DWORD cItems;
3. [size\_is(cItems)] DS\_DOMAIN\_CONTROLLER\_INFO\_FFFFFFFFW\* rItems;
4. } DRS\_MSG\_DCINFOREPLY\_VFFFFFFFF;

**cItems:**  The count of items in the **rItems** array.

**rItems:**  Information about the active [**LDAP connections**](#gt_198f4791-cea3-465d-89e2-262991624e08).

##### DS\_DOMAIN\_CONTROLLER\_INFO\_1W

The DS\_DOMAIN\_CONTROLLER\_INFO\_1W structure defines [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) information that is returned as a part of the response to an InfoLevel = 1 request. The struct contains information about a single DC in the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

1. typedef struct {
2. [string, unique] WCHAR\* NetbiosName;
3. [string, unique] WCHAR\* DnsHostName;
4. [string, unique] WCHAR\* SiteName;
5. [string, unique] WCHAR\* ComputerObjectName;
6. [string, unique] WCHAR\* ServerObjectName;
7. BOOL fIsPdc;
8. BOOL fDsEnabled;
9. } DS\_DOMAIN\_CONTROLLER\_INFO\_1W;

**NetbiosName:**  NetBIOS name of the DC.

**DnsHostName:**  DNS host name of the DC.

**SiteName:**  [**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) of the site [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

**ComputerObjectName:**  [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the computer object that corresponds to the DC.

**ServerObjectName:**  DN of the server object that corresponds to the DC.

**fIsPdc:**  True if and only if the DC is the [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) [**FSMO role owner**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b).

**fDsEnabled:**  A Boolean value that indicates whether or not the machine is a domain controller. This value MUST be TRUE.

##### DS\_DOMAIN\_CONTROLLER\_INFO\_2W

The DS\_DOMAIN\_CONTROLLER\_INFO\_2W structure defines [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) information that is returned as a part of the response to an InfoLevel = 2 request. The struct contains information about a single DC in the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

1. typedef struct {
2. [string, unique] WCHAR\* NetbiosName;
3. [string, unique] WCHAR\* DnsHostName;
4. [string, unique] WCHAR\* SiteName;
5. [string, unique] WCHAR\* SiteObjectName;
6. [string, unique] WCHAR\* ComputerObjectName;
7. [string, unique] WCHAR\* ServerObjectName;
8. [string, unique] WCHAR\* NtdsDsaObjectName;
9. BOOL fIsPdc;
10. BOOL fDsEnabled;
11. BOOL fIsGc;
12. GUID SiteObjectGuid;
13. GUID ComputerObjectGuid;
14. GUID ServerObjectGuid;
15. GUID NtdsDsaObjectGuid;
16. } DS\_DOMAIN\_CONTROLLER\_INFO\_2W;

**NetbiosName:**  NetBIOS name of the DC.

**DnsHostName:**  DNS host name of the DC.

**SiteName:**  [**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) of the site [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

**SiteObjectName:**  [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the site object.

**ComputerObjectName:**  DN of the computer object that corresponds to the DC.

**ServerObjectName:**  DN of the server object that corresponds to the DC.

**NtdsDsaObjectName:**  DN of the nTDSDSA object that corresponds to the DC.

**fIsPdc:**  True if and only if the DC is the [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) [**FSMO role owner**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b).

**fDsEnabled:**  A Boolean value that indicates whether or not the machine is a domain controller. This value MUST be TRUE.

**fIsGc:**  True if and only if the DC is also a [**GC**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169).

**SiteObjectGuid:**  The objectGUID [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of the site object.

**ComputerObjectGuid:**  The objectGUID attribute of the computer object that corresponds to the DC.

**ServerObjectGuid:**  The objectGUID attribute of the server object that corresponds to the DC.

**NtdsDsaObjectGuid:**  The objectGUID attribute of the nTDSDSA object that corresponds to the DC.

##### DS\_DOMAIN\_CONTROLLER\_INFO\_3W

The DS\_DOMAIN\_CONTROLLER\_INFO\_3W structure defines [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) information that is returned as a part of the response to an InfoLevel = 3 request. The struct contains information about a single DC in the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

1. typedef struct {
2. [string, unique] WCHAR\* NetbiosName;
3. [string, unique] WCHAR\* DnsHostName;
4. [string, unique] WCHAR\* SiteName;
5. [string, unique] WCHAR\* SiteObjectName;
6. [string, unique] WCHAR\* ComputerObjectName;
7. [string, unique] WCHAR\* ServerObjectName;
8. [string, unique] WCHAR\* NtdsDsaObjectName;
9. BOOL fIsPdc;
10. BOOL fDsEnabled;
11. BOOL fIsGc;
12. BOOL fIsRodc;
13. GUID SiteObjectGuid;
14. GUID ComputerObjectGuid;
15. GUID ServerObjectGuid;
16. GUID NtdsDsaObjectGuid;
17. } DS\_DOMAIN\_CONTROLLER\_INFO\_3W;

**NetbiosName:**  NetBIOS name of the DC.

**DnsHostName:**  DNS host name of the DC.

**SiteName:**  [**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) of the [**site object**](#gt_0ce6abc5-9823-4a69-bb30-12e42ff99629).

**SiteObjectName:**  [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the site object.

**ComputerObjectName:**  DN of the [**computer object**](#gt_d8e8f5a7-db85-40a8-98ed-1abab2237b82) that corresponds to the DC.

**ServerObjectName:**  DN of the [**server object**](#gt_62a8c543-5998-480b-8fa7-41a8f04a18e5) that corresponds to the DC.

**NtdsDsaObjectName:**  DN of the [**nTDSDSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8) that corresponds to the DC.

**fIsPdc:**  True if and only if the DC is the [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) [**FSMO role owner**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b).

**fDsEnabled:**  A Boolean value that indicates whether or not the machine is a domain controller. This value MUST be TRUE.

**fIsGc:**  True if and only if the DC is also a [**GC**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169).

**fIsRodc:**  True if and only if the DC is an [**RODC**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870).

**SiteObjectGuid:**  objectGUID of the site object.

**ComputerObjectGuid:**  objectGUID of the computer object that corresponds to the DC.

**ServerObjectGuid:**  objectGUID of the server object that corresponds to the DC.

**NtdsDsaObjectGuid:**  objectGUID of the nTDSDSA object that corresponds to the DC.

##### DS\_DOMAIN\_CONTROLLER\_INFO\_FFFFFFFFW

The DS\_DOMAIN\_CONTROLLER\_INFO\_FFFFFFFFW structure defines [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) information that is returned as a part of the response to an InfoLevel = 0xFFFFFFFF request. The struct contains information about a single [**LDAP connection**](#gt_198f4791-cea3-465d-89e2-262991624e08) to the current server.

1. typedef struct {
2. DWORD IPAddress;
3. DWORD NotificationCount;
4. DWORD secTimeConnected;
5. DWORD Flags;
6. DWORD TotalRequests;
7. DWORD Reserved1;
8. [string, unique] WCHAR\* UserName;
9. } DS\_DOMAIN\_CONTROLLER\_INFO\_FFFFFFFFW;

**IPAddress:**  The IPv4 address of the client that established the LDAP connection to the server. If the client is connected with IPv6, this field MUST be zero.

**NotificationCount:**  Number of [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) notifications enabled on the server.

**secTimeConnected:**  Total time in number of seconds that the connection is established.

**Flags:**  Zero or more of the bit flags from [LDAP\_CONN\_PROPERTIES](#Section_09a9cd41caed441da7515a992800a4fb) indicating the properties of this connection.

**TotalRequests:**  Total number of LDAP requests made on this LDAP connection.

**Reserved1:**  Unused. MUST be 0 and ignored.

**UserName:**  Name of the [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) that established the LDAP connection.

#### Server Behavior of the IDL\_DRSDomainControllerInfo Method

*Informative summary of behavior*: The [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) method supports four information levels. For levels 1, 2, and 3, the server returns information for the [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) in the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) of the server. For level 0xffffffff, the server returns information about the [**LDAP connections**](#gt_198f4791-cea3-465d-89e2-262991624e08) on the server that are currently open.

Regular read access checks apply to the information that is returned to the caller. Therefore, if the caller does not have [**read permission**](#gt_e5834747-516f-4142-ae27-cafb41ee9fd6) on data that needs to be returned, this data is not included in the response. See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.4.3 for more information about access check behavior in read operations.

For information about the Windows versions in which information levels were introduced and supported, see the following behavior note.[<14>](#Appendix_A_14" \o "Product behavior note 14)

**Note**  The server behavior of the IDL\_DRSDomainControllerInfo method uses the [CrackNames](#Section_36653ac4f0e243b79f158fb44850945a) procedure defined in section 4.1.4.2.9.

1. ULONG
2. IDL\_DRSDomainControllerInfo(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_DCINFOREQ \*pmsgIn,
6. [out, ref] DWORD \*pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)] DRS\_MSG\_DCINFOREPLY \*pmsgOut)
8. msgIn: DRS\_MSG\_DCINFOREQ\_V1
9. infoLevel, i: integer
10. domainName: unicodestring
11. dcSet: set of DSName
12. serversContainer, crObj, dcObj, dsaObj, svrObj, siteObj, obj, v: DSName
13. lc: DS\_DOMAIN\_CONTROLLER\_INFO\_FFFFFFFFW
14. rI1: ADDRESS OF DS\_DOMAIN\_CONTROLLER\_INFO\_1W
15. rI2: ADDRESS OF DS\_DOMAIN\_CONTROLLER\_INFO\_2W
16. rI3: ADDRESS OF DS\_DOMAIN\_CONTROLLER\_INFO\_3W
17. found: boolean
18. crackMsgIn: DRS\_MSG\_CRACKREQ\_V1
19. crackOut: DS\_NAME\_RESULTW
20. outV: DWORD
21. userAccountControl: set of integer
22. ValidateDRSInput(hDrs, 16)
23. msgIn := pmsgIn^.V1
24. infoLevel := msgIn.InfoLevel
25. domainName := msgIn.Domain
26. pdwOutVersion^ := infoLevel
27. if infoLevel = 1 then
28. pmsgOut^.V1.cItems := 0
29. pmsgOut^.V1.rItems := null
30. else if infoLevel = 2 then
31. pmsgOut^.V2.cItems := 0
32. pmsgOut^.V2.rItems := null
33. else if infoLevel = 3 then
34. pmsgOut^.V3.cItems := 0
35. pmsgOut^.V3.rItems := null
36. else if infoLevel = 0xFFFFFFFF then
37. pmsgOut^.VFFFFFFFF.cItems := 0
38. pmsgOut^.VFFFFFFFF.rItems := null
39. endif
40. if dwInVersion ≠ 1 then
41. return ERROR\_INVALID\_PARAMETER
42. endif
43. if not (infoLevel in {1,2,3,0xFFFFFFFF}) then
44. return ERROR\_INVALID\_PARAMETER
45. endif
46. if infoLevel = 0xFFFFFFFF then
47. /\* Enumerate the LDAP connections. \*/
48. if not IsMemberOfBuiltinAdminGroup() then
49. return ERROR\_ACCESS\_DENIED
50. endif
51. pmsgOut^.VFFFFFFFF.cItems := number(dc.ldapConnections)
52. i := 0
53. foreach lc in dc.ldapConnections
54. pmsgOut^.VFFFFFFFF.rItems[i].IPAddress := lc.iPAddress
55. pmsgOut^.VFFFFFFFF.rItems[i].NotificationCount :=
56. lc.notificationCount
57. pmsgOut^.VFFFFFFFF.rItems[i].secTimeConnected :=
58. lc.secTimeConnected
59. pmsgOut^.VFFFFFFFF.rItems[i].Flags := lc.flags
60. pmsgOut^.VFFFFFFFF.rItems[i].TotalRequests := lc.totalRequests
61. pmsgOut^.VFFFFFFFF.rItems[i].UserName := lc.userName
62. pmsgOut^.VFFFFFFFF.rItems[i].Reserved1 := 0
63. i := i + 1
64. endfor
65. return 0
66. endif
67. /\* Verify that the given domain name matches the default domain NC.
68. \* First check if it is the nETBiosName or dNSHostName of the default
69. \* domain NC by searching for the crossRef object. If this doesn't
70. \* find a match, call IDL\_DRSCrackNames to check if the given
71. \* domain name is a name for the default domain NC. \*/
72. crObj := select one v from children
73. DescendantObject(ConfigNC(), "CN=Partitions,")
74. where
75. (v!dnsRoot = domainName or v!nETBiosName = domainName)
76. and
77. v!nCName = DefaultNC()
78. found := (crObj ≠ null)
79. if not found then
80. /\* Not found; use IDL\_DRSCrackNames to resolve the name. \*/
81. crackMsgIn.dwFlags := 0
82. crackMsgIn.formatOffered := DS\_UNKNOWN\_NAME
83. crackMsgIn.formatDesired := DS\_FQDN\_1779\_NAME
84. crackMsgIn.cNames := 3
85. crackMsgIn.rpNames[0] := domainName
86. crackMsgIn.rpNames[1] := domainName + "\"
87. crackMsgIn.rpNames[2] := domainName + "/"
88. /\* Call IDL\_DRSCrackNames as a local procedure. \*/
89. CrackNames(crackMsgIn, ADR(crackOut))
91. i := 0
92. while i < 3 and not found
93. if crackOut.rItems[i].status = DS\_NAME\_NO\_ERROR
94. then
95. if crackOut.rItems[i].pName = DefaultNC().dn
96. then
97. found := true
98. else
99. return ERROR\_INVALID\_PARAMETER
100. endif
101. endif
102. i := i + 1
103. endwhile
104. endif
105. if not found then
106. return ERROR\_DS\_OBJ\_NOT\_FOUND
107. endif
108. /\* Enumerate the DCs in the domain. \*/
109. if infoLevel = 3 then
110. /\* client requests to return RODCs too \*/
111. userAccountControl :=
112. {ADS\_UF\_SERVER\_TRUST\_ACCOUNT, ADS\_UF\_PARTIAL\_SECRETS\_ACCOUNT}
113. else
114. userAccountControl := {ADS\_UF\_SERVER\_TRUST\_ACCOUNT}
115. endif
116. dcSet := select all v from subtree DefaultNC() where
117. v!objectCategory = GetDefaultObjectCategory(computer)
118. and (userAccountControl ∩ v!userAccountControl ≠ null)
119. if infoLevel = 1 then
120. pmsgOut^.V1.cItems := number(dcSet)
121. i := 0
122. foreach dcObj in dcSet
123. rI1 := ADR(pmsgOut^.V1.rItems[i])
124. rI1^.DnsHostName := dcObj!dNSHostName
125. rI1^.ComputerObjectName := dcObj.dn
126. /\* sAMAccountName excluding the "$" at the end. \*/
127. rI1^.NetbiosName := SubString(dcObj!sAMAccountName, 0,
128. dcObj!samAccountName.length-1)
129. rI1^.fDsEnabled := true
130. /\* select a server object from the serverReferenceBL, it is
131. preferred that the server object has a child object with
132. CN "NTDS Settings" \*/
133. svrObj :=
134. select one v from all where v.dn in dcObj!serverReferenceBL
135. and DescendantObject(v, "CN=NTDS Settings") ≠ null
136. if svrObj = null then
137. svrObj :=
138. select one v from all where v.dn in dcObj!serverReferenceBL
139. endif
140. if svrObj ≠ null then
141. rI1^.ServerObjectName := svrObj.dn
142. serversContainer :=
143. select one o from all where o!objectGUID = svrObj!parent
144. siteObj := serversContainer!parent
145. rI1^.SiteObjectName := siteObj.dn
146. dsaObj := DescendantObject(v, "CN=NTDS Settings,")
147. rI1^.fIsPdc := (dsaObj = GetFSMORoleOwner(FSMO\_PDC))
148. endif
149. i := i + 1
150. endfor
151. else
152. if infoLevel = 2 then
153. pmsgOut^.V2.cItems := number(dcSet)
154. i := 0
155. foreach dcObj in dcSet
156. rI2 := ADR(pmsgOut^.V2.rItems[i])
157. rI2^.DnsHostName := dcObj!dNSHostName
158. rI2^.ComputerObjectName := dcObj.dn
159. /\* sAMAccountName excluding the "$" at the end. \*/
160. rI2^.NetbiosName := SubString(dcObj!samAccountName, 0,
161. dcObj!samAccountName.length-1)
162. rI2^.ComputerObjectGUID := dcObj.guid
163. rI2^.fDsEnabled := true
164. /\* select a server object from the serverReferenceBL, it is
165. preferred that the server object has a child object with
166. CN "NTDS Settings" \*/
167. svrObj :=
168. select one v from all where v.dn in dcObj!serverReferenceBL
169. and DescendantObject(v, "CN=NTDS Settings") ≠ null
170. if svrObj = null then
171. svrObj :=
172. select one v from all where v.dn in dcObj!serverReferenceBL
173. endif
174. if svrObj ≠ null then
175. rI2^.ServerObjectName := svrObj.dn
176. rI2^.ServerObjectGuid := svrObj.guid
177. serversContainer :=
178. select one o from all where o!objectGUID = svrObj!parent
179. siteObj := serversContainer!parent
180. rI2^.SiteObjectName := siteObj.dn
181. rI2^.SiteObjectGUID := siteObj.guid
182. dsaObj := DescendantObject(v, "CN=NTDS Settings,")
183. rI2^.NtdsDsaObjectGUID := dsaObj.guid
184. rI2^.fIsGc := (NTDSDSA\_OPT\_IS\_GC in dsaObj!options)
185. rI2^.fIsPdc := (dsaObj = GetFSMORoleOwner(FSMO\_PDC))
186. endif
187. i := i + 1
188. endfor
189. else
190. /\* infoLevel = 3 \*/
191. pmsgOut^.V3.cItems := number(dcSet)
192. i := 0
193. foreach dcObj in dcSet
194. rI3 := ADR(pmsgOut^.V3.rItems[i])
195. rI3^.DnsHostName := dcObj!dNSHostName
196. rI3^.ComputerObjectName := dcObj.dn
197. /\* sAMAccountName excluding the "$" at the end. \*/
198. rI3^.NetbiosName := SubString(dcObj!samAccountName, 0,
199. dcObj!samAccountName.length-1)
200. rI3^.ComputerObjectGUID := dcObj.guid
201. rI3^.fDsEnabled := true
202. /\* select a server object from the serverReferenceBL, it is
203. preferred that the server object has a child object with
204. CN "NTDS Settings" \*/
205. svrObj :=
206. select one v from all where v.dn in dcObj!serverReferenceBL
207. and DescendantObject(v, "CN=NTDS Settings") ≠ null
208. if svrObj = null then
209. svrObj :=
210. select one v from all where v.dn in dcObj!serverReferenceBL
211. endif
212. if svrObj ≠ null then
213. rI3^.ServerObjectName := svrObj.dn
214. rI3^.ServerObjectGuid := svrObj.guid
215. serversContainer :=
216. select one o from all where o!objectGUID = svrObj!parent
217. siteObj := serversContainer!parent
218. rI3^.SiteObjectName := siteObj.dn
219. rI3^.SiteObjectGUID := siteObj.guid
220. dsaObj := DescendantObject(v, "CN=NTDS Settings,")
221. rI3^.NtdsDsaObjectGUID := dsaObj.guid
222. rI3^.fIsGC := (NTDSDSA\_OPT\_IS\_GC in dsaObj!options)
223. rI3^.fIsPDC := (dsaObj = GetFSMORoleOwner(FSMO\_PDC))
224. rI3^.fIsRodc := ((ADS\_UF\_PARTIAL\_SECRETS\_ACCOUNT ∩
225. dcObj!userAccountControl) ≠ null)
226. endif
227. i := i + 1
228. endfor
229. endif
230. endif
231. return 0

#### Examples of the IDL\_DRSDomainControllerInfo Method

An application running on DC2 invokes the [DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) method on DC2 to retrieve the NetBIOS and DNS host names for all [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) in the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) CONTOSO.COM.

##### Initial State

Querying the crossRef [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) CONTOSO.COM on DC2 by performing an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) search with base scope on the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) 'CN=CONTOSO,CN=Partitions,CN=Configuration,DC=contoso,DC=com':

* Expanding base 'CN=CONTOSO,CN=Partitions,CN=Configuration,DC=contoso,DC=com'...
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=CONTOSO,CN=Partitions,CN=Configuration,DC=contoso,DC=com
  + 2> objectClass: top; crossRef;
  + 1> nCName: DC=contoso,DC=com;
  + 1> dnsRoot: contoso.com;
  + 1> nETBIOSName: CONTOSO;

Querying the DC1 computer object in domain NC DC=CONTOSO, DC=COM by performing an LDAP search with base scope on the DN 'CN=DC1,OU=Domain Controllers,DC=contoso,DC=com':

* Expanding base 'CN=DC1,OU=Domain Controllers,DC=contoso,DC=com'...
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=DC1,OU=Domain Controllers,DC=contoso,DC=com
  + 5> objectClass: top; person; organizationalPerson; user; computer;
  + 1> cn: DC1;
  + 1> distinguishedName: CN=DC1, OU=Domain Controllers, DC=contoso, DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/10/2006 18:04:35 Pacific Standard Daylight Time;
  + 1> whenChanged: 07/15/2006 19:39:05 Pacific Standard Daylight Time;
  + 1> uSNCreated: 12291;
  + 1> uSNChanged: 24577;
  + 1> name: DC1;
  + 1> objectGUID: ac1993e1-0377-4161-893e-ccd2a98e1bba;
  + 1> userAccountControl: (UF\_SERVER\_TRUST\_ACCOUNT | UF\_TRUSTED\_FOR\_DELEGATION );
  + 1> badPwdCount: 0;
  + 1> codePage: 0;
  + 1> countryCode: 0;
  + 1> badPasswordTime: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogoff: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogon: 07/17/2006 19:47:40 Pacific Standard Daylight Time;
  + 1> localPolicyFlags: 0;
  + 1> pwdLastSet: 07/10/2006 18:04:35 Pacific Standard Daylight Time;
  + 1> primaryGroupID: 516;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1001;
  + 1> accountExpires: 09/14/30828 02:48:05 UNC ;
  + 1> logonCount: 17;
  + 1> sAMAccountName: DC1$;
  + 1> sAMAccountType: 805306369;
  + 1> operatingSystem: Windows Server 2003 operating system;
  + 1> operatingSystemVersion: 5.2 (3790);
  + 1> operatingSystemServicePack: Service Pack 1;
  + 1> serverReferenceBL: CN=DC1,CN=Servers, CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dNSHostName: DC1.contoso.com;
  + 1> rIDSetReferences: CN=RID Set,CN=DC1,OU=Domain Controllers, DC=contoso, DC=com;
  + 15> servicePrincipalName: ldap/DC1.contoso.com/NDNC5.contoso.com; ldap/DC1.contoso.com/NDNC2.contoso.com; ldap/DC1.contoso.com/NDNC1.contoso.com; GC/DC1.contoso.com/contoso.com; HOST/DC1.contoso.com/CONTOSO; HOST/DC1; HOST/DC1.contoso.com; HOST/DC1.contoso.com/contoso.com; E3514235-4B06-11D1-AB04-00C04FC2DCD2/c20bc312-4d35-4cc0-9903-b1073368af4a/contoso.com; ldap/c20bc312-4d35-4cc0-9903-b1073368af4a.\_msdcs.contoso.com; ldap/DC1.contoso.com/CONTOSO; ldap/DC1; ldap/DC1.contoso.com; ldap/DC1.contoso.com/contoso.com; NtFrs-88f5d2bd-b646-11d2-a6d3-00c04fc9b232/DC1.contoso.com;
  + 1> objectCategory: CN=Computer, CN=Schema, CN=Configuration, DC=contoso, DC=com;
  + 1> isCriticalSystemObject: TRUE;
  + 1> frsComputerReferenceBL: CN=DC1, CN=Domain System Volume (SYSVOL share),CN=File Replication Service,CN=System,DC=contoso,DC=com;
  + 1> lastLogonTimestamp: 07/11/2006 04:02:42 Pacific Std Daylight Time;

Querying the DC1 computer object in domain NC DC=CONTOSO, DC=COM by performing an LDAP search with base scope on the DN 'CN=DC2,OU=Domain Controllers,DC=contoso,DC=com':

* Expanding base 'CN=DC2,OU=Domain Controllers,DC=contoso,DC=com'...
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=DC2,OU=Domain Controllers,DC=contoso,DC=com
  + 5> objectClass: top; person; organizationalPerson; user; computer;
  + 1> cn: DC2;
  + 1> distinguishedName: CN=DC2, OU=Domain Controllers, DC=contoso, DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/10/2006 18:12:01 Pacific Standard Daylight Time;
  + 1> whenChanged: 07/16/2006 13:46:14 Pacific Standard Daylight Time;
  + 1> displayName: DC2$;
  + 1> uSNCreated: 13711;
  + 1> uSNChanged: 28819;
  + 1> name: DC2;
  + 1> objectGUID: 09697f46-2458-4b26-a4e9-aa36059421c4;
  + 1> userAccountControl: (UF\_SERVER\_TRUST\_ACCOUNT | UF\_TRUSTED\_FOR\_DELEGATION );
  + 1> badPwdCount: 0;
  + 1> codePage: 0;
  + 1> countryCode: 0;
  + 1> badPasswordTime: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogoff: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogon: 07/17/2006 20:38:08 Pacific Standard Daylight Time;
  + 1> localPolicyFlags: 0;
  + 1> pwdLastSet: 07/10/2006 18:12:02 Pacific Standard Daylight Time;
  + 1> primaryGroupID: 516;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1102;
  + 1> accountExpires: 09/14/30828 02:48:05 UNC ;
  + 1> logonCount: 8;
  + 1> sAMAccountName: DC2$;
  + 1> sAMAccountType: 805306369;
  + 1> operatingSystem: Windows Server 2003;
  + 1> operatingSystemVersion: 5.2 (3790);
  + 1> operatingSystemServicePack: Service Pack 1;
  + 1> serverReferenceBL: CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dNSHostName: DC2.contoso.com;
  + 1> rIDSetReferences: CN=RID Set,CN=DC2,OU=Domain Controllers, DC=contoso, DC=com;
  + 14> servicePrincipalName: ldap/DC2.contoso.com/NDNC5.contoso.com; ldap/DC2.contoso.com/NDNC2.contoso.com; ldap/6aad8f5a-07cc-403a-9696-9102fe1c320b.\_msdcs.contoso.com; ldap/DC2.contoso.com/CONTOSO; ldap/DC2; ldap/DC2.contoso.com; ldap/DC2.contoso.com/contoso.com; NtFrs-88f5d2bd-b646-11d2-a6d3-00c04fc9b232/DC2.contoso.com; HOST/DC2.contoso.com/CONTOSO; HOST/DC2.contoso.com/contoso.com; C/DC2.contoso.com/contoso.com; E3514235-4B06-11D1-AB04-00C04FC2DCD2/6aad8f5a-07cc-403a-9696-9102fe1c320b/contoso.com;
  + HOST/DC2; HOST/DC2.contoso.com;
  + 1> objectCategory: CN=Computer, CN=Schema, CN=Configuration, DC=contoso, DC=com;
  + 1> isCriticalSystemObject: TRUE;
  + 1> frsComputerReferenceBL: CN=DC2,CN=Domain System Volume (SYSVOL share),CN=File Replication Service,CN=System,DC=contoso,DC=com;
  + 4> dSCorePropagationData: 07/10/2006 18:14:51 Pacific Standard Daylight Time; 07/10/2006 18:14:51 Pacific Standard Time Pacific Daylight Time; 07/10/2006 18:14:51 Pacific Standard Time Pacific Daylight Time; 01/08/1601 07:15:13 Pacific Standard Time Pacific Daylight Time;
  + 1> lastLogonTimestamp: 07/10/2006 19:52:48 Pacific Std Daylight Time;

##### Client Request

DC2 invokes the [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736) method against itself with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC2 is omitted):

* *dwInVersion* = 1
* *pmsgIn* = [DRS\_MSG\_DCINFOREQ\_V1](#Section_18b23122a1c24367a677592e0d4eef18)
  + Domain = "contoso.com"
  + InfoLevel = 1

##### Server Response

Return code of 0 and the following values:

* *pdwOutVersion^* = 1
* *pmsgOut* = [DRS\_MSG\_DCINFOREPLY\_V1](#Section_f71a8f6c54264628aa91aeabef2c086f)
  + cItems: 2
  + rItems[0]: [DS\_DOMAIN\_CONTROLLER\_INFO\_1W](#Section_b30c5951ccb14fb6ba9a5699d5d78759)
    - NetbiosName: "DC1"
    - DnsHostName: "DC1.contoso.com"
    - SiteName: "Default-First-Site-Name"
    - ComputerObjectName: "CN=DC1, OU=Domain Controllers,DC=contoso,DC=com"
    - ServerObjectName: "CN=DC1,CN=Servers, CN=Default-First-Site-Name,CN=Sites, CN=Configuration, DC=contoso,DC=com"
    - fIsPdc: 1
    - fDsEnabled: 1
  + rItems[1]: DS\_DOMAIN\_CONTROLLER\_INFO\_1W
    - NetbiosName: "DC2"
    - DnsHostName: "DC2.contoso.com"
    - SiteName: "Default-First-Site-Name"
    - ComputerObjectName: "CN=DC2, OU=Domain Controllers,DC=contoso,DC=com"
    - ServerObjectName: "CN=DC2,CN=Servers, CN=Default-First-Site-Name,CN=Sites, CN=Configuration, DC=contoso,DC=com"
    - fIsPdc: 0
    - fDsEnabled: 1

##### Final State

The final state is the same as the initial state; there is no change.

### IDL\_DRSExecuteKCC (Opnum 18)

The IDL\_DRSExecuteKCC method validates the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) interconnections of [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) and [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) them if necessary.

1. ULONG IDL\_DRSExecuteKCC(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_KCC\_EXECUTE\* pmsgIn
6. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_KCC\_EXECUTE

The DRS\_MSG\_KCC\_EXECUTE union defines the request messages sent to the [IDL\_DRSExecuteKCC](#Section_ad807917687b40d9abe2053af0246523) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_KCC\_EXECUTE\_V1 V1;
6. } DRS\_MSG\_KCC\_EXECUTE;

**V1:**  Version 1 request.

##### DRS\_MSG\_KCC\_EXECUTE\_V1

The DRS\_MSG\_KCC\_EXECUTE\_V1 structure defines the request message sent to the [IDL\_DRSExecuteKCC](#Section_ad807917687b40d9abe2053af0246523) method.

1. typedef struct {
2. DWORD dwTaskID;
3. DWORD dwFlags;
4. } DRS\_MSG\_KCC\_EXECUTE\_V1;

**dwTaskID:**  MUST be 0.

**dwFlags:**  Zero or more of the following bit flags, which are presented in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | X | X | D P | A S | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**AS (DS\_KCC\_FLAG\_ASYNC\_OP, 0x00000001)**: Request the [**KCC**](#gt_c7d4f1f6-5285-4168-b21a-022f775a3f58) to run, then return immediately.

**DP (DS\_KCC\_FLAG\_DAMPED, 0x00000002)**: Request the KCC to run unless there is already such a request pending according to implementation-defined rules. Implementations MAY choose to ignore this flag and always request the KCC to run.

#### Method-Specific Abstract Types and Procedures

##### ExecuteKCCTasks

1. procedure ExecuteKCCTasks(): ULONG

This procedure executes the tasks necessary for maintaining the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) topology between [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). These tasks include activities such as maintenance of kCCFailedLinks and kCCFailedConnections, maintenance of intrasite and intersite connections, and [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) of [**RODC**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870) [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) (as appropriate). See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.2.2 for a full list of these tasks.

If an error occurs, a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) is returned. If successful, the method returns 0.

#### Server Behavior of the IDL\_DRSExecuteKCC Method

*Informative summary of behavior*: The [IDL\_DRSExecuteKCC](#Section_ad807917687b40d9abe2053af0246523) method triggers the execution of tasks that generate and maintain the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) topology between [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).[<15>](#Appendix_A_15" \o "Product behavior note 15) See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.2.2 for more information related to the tasks performed by the [**KCC**](#gt_c7d4f1f6-5285-4168-b21a-022f775a3f58) upon receipt of an IDL\_DRSExecuteKCC request.

1. ULONG
2. IDL\_DRSExecuteKCC(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_KCC\_EXECUTE \*pmsgIn)
7. msgIn: DRS\_MSG\_KCC\_EXECUTE\_V1
8. ValidateDRSInput(hDrs, 18)
9. /\* Validate the request version \*/
10. if dwInVersion ≠ 1 then
11. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
12. endif
13. msgIn := pmsgIn^.V1
14. if msgIn.dwTaskID ≠ 0 then
15. return ERROR\_INVALID\_PARAMETER
16. endif
17. if not AccessCheckCAR(ConfigNC(), DS-Replication-Manage-Topology)
18. then
19. return ERROR\_DS\_DRA\_ACCESS\_DENIED
20. endif
21. if msgIn.dwFlags = DS\_KCC\_FLAG\_ASYNC\_OP then
22. Asynchronous Processing: Initiate a logical thread of control
23. to process the remainder of this request asynchronously
24. return 0
25. endif
26. return ExecuteKCCTasks()

### IDL\_DRSFinishDemotion (Opnum 27)

The IDL\_DRSFinishDemotion method either performs one or more steps toward the complete removal of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) from an [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62), or it undoes the effects of the first phase of removal (performed by [IDL\_DRSInitDemotion](#Section_faca17da3f7f409298dbfd2ce7d98b8c)). This method is supported by AD LDS only.

1. ULONG IDL\_DRSFinishDemotion(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_FINISH\_DEMOTIONREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_FINISH\_DEMOTIONREPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_FINISH\_DEMOTIONREQ

The DRS\_MSG\_FINISH\_DEMOTIONREQ union defines the request messages sent to the [IDL\_DRSFinishDemotion](#Section_0bf530e81be04f48b8c2208031a8725f) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_FINISH\_DEMOTIONREQ\_V1 V1;
6. } DRS\_MSG\_FINISH\_DEMOTIONREQ;

**V1:**  Version 1 request. Currently, only one version is defined.

##### DRS\_MSG\_FINISH\_DEMOTIONREQ\_V1

The DRS\_MSG\_FINISH\_DEMOTIONREQ\_V1 structure defines the request message sent to the [IDL\_DRSFinishDemotion](#Section_0bf530e81be04f48b8c2208031a8725f) method.

1. typedef struct {
2. DWORD dwOperations;
3. UUID uuidHelperDest;
4. [string] LPWSTR szScriptBase;
5. } DRS\_MSG\_FINISH\_DEMOTIONREQ\_V1;

**dwOperations:**  Zero or more of the following bit flags, which are presented in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | U 2 | U 1 | D | C | R | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | F | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**R (DS\_DEMOTE\_ROLLBACK\_DEMOTE, 0x00000001)**: Undo the effects of [IDL\_DRSInitDemotion](#Section_faca17da3f7f409298dbfd2ce7d98b8c). If present, any other flags present (except for DS\_DEMOTE\_OPT\_FAIL\_ON\_UNKNOWN) are ignored.

**C (DS\_DEMOTE\_COMMIT\_DEMOTE, 0x00000002)**: Discontinue being a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) for the current DC instance by stopping all [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) protocols.

**D (DS\_DEMOTE\_DELETE\_CSMETA, 0x00000004)**: Delete the nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for this DC; see [RemoveADLDSServer (section 4.1.7.2.1)](#Section_9bb77bbf7fcc40a99cb9a60739e3cf41).

**U1 (DS\_DEMOTE\_UNREGISTER\_SCPS, 0x00000008)**: Delete any serviceConnectionPoint objects for this DC from [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024); see [RemoveADLDSSCP (section 4.1.7.2.2)](#Section_1e458dcf4e99480ba543a03ea0c65403).

**U2 (DS\_DEMOTE\_UNREGISTER\_SPNS, 0x00000010)**: Delete any AD LDS [**SPNs**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) from the object (in the external AD DS [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca)) that corresponds to the [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) that the AD LDS service is running as; see [RemoveADLDSSPNs (section 4.1.7.2.3)](#Section_3090a83567d9404a81109d443a0c8921).

**F (DS\_DEMOTE\_OPT\_FAIL\_ON\_UNKNOWN\_OP, 0x80000000)**: If this flag is present, then the request fails.

**uuidHelperDest:** Unused. Must be [**NULL GUID**](#gt_ba500a5b-8c29-467c-a335-0980c8b11304) and ignored.

**szScriptBase:** The path name of the folder in which to store SPN unregistration scripts. Required when DS\_DEMOTE\_UNREGISTER\_SPNS is specified in **dwOperations**.

##### DRS\_MSG\_FINISH\_DEMOTIONREPLY

The DRS\_MSG\_FINISH\_DEMOTIONREPLY union defines the response messages received from the [IDL\_DRSFinishDemotion](#Section_0bf530e81be04f48b8c2208031a8725f) method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_FINISH\_DEMOTIONREPLY\_V1 V1;
6. } DRS\_MSG\_FINISH\_DEMOTIONREPLY;

**V1:**  Version 1 reply.

##### DRS\_MSG\_FINISH\_DEMOTIONREPLY\_V1

The DRS\_MSG\_FINISH\_DEMOTIONREPLY\_V1 structure defines the response message received from the [IDL\_DRSFinishDemotion](#Section_0bf530e81be04f48b8c2208031a8725f) method.

1. typedef struct {
2. DWORD dwOperationsDone;
3. DWORD dwOpFailed;
4. DWORD dwOpError;
5. } DRS\_MSG\_FINISH\_DEMOTIONREPLY\_V1;

**dwOperationsDone:**  The set of operations that were successfully performed. This can include the following values: DS\_DEMOTE\_ROLLBACK\_DEMOTE, DS\_DEMOTE\_COMMIT\_DEMOTE, DS\_DEMOTE\_DELETE\_CSMETA, DS\_DEMOTE\_UNREGISTER\_SCPS, DS\_DEMOTE\_UNREGISTER\_SPNS. This MUST include any value from the input element DRS\_MSG\_FINISH\_DEMOTIONREQ\_V1.dwOperations whose corresponding operations (see pseudocode in section [4.1.7.3](#Section_cbed0810e977471b9abc2573014f7c5b)) succeeded.

**dwOpFailed:**  The set of operations that failed during demotion. This can include the same values as the **dwOperationsDone** field. This MUST include any value from the input element DRS\_MSG\_FINISH\_DEMOTIONREQ\_V1.dwOperations whose corresponding operations (see pseudocode in section 4.1.7.3) failed.

**dwOpError:**  The Win32 error code (as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.2) of the first failed operation (if any), from the following operations: DS\_DEMOTE\_ROLLBACK\_DEMOTE, DS\_DEMOTE\_COMMIT\_DEMOTE, DS\_DEMOTE\_DELETE\_CSMETA, or DS\_DEMOTE\_UNREGISTER\_SCPS.

#### Method-Specific Abstract Types and Procedures

##### RemoveADLDSServer

1. procedure RemoveADLDSServer(): DWORD

The RemoveADLDSServer procedure connects to any available [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) partner and uses the [IDL\_DRSRemoveDsServer](#Section_d5c310ae347a49d4a78e6ffb2eecd581) method to delete the nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that corresponds to this [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). If no replication partner is available, or if a replication partner is available and either no such nTDSDSA object exists or the deletion is successful, RemoveADLDSServer returns ERROR\_SUCCESS; otherwise, it returns a Win32 error.

##### RemoveADLDSSCP

1. procedure RemoveADLDSSCP(): DWORD

The RemoveADLDSSCP procedure connects to an [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024) [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) and deletes any serviceConnectionPoint [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that was created in AD DS for this [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) DC. See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.3.8 for more details on AD LDS serviceConnectionPoint objects. If no such serviceConnectionPoint object exists or if the deletion is successful, RemoveADLDSSCP returns ERROR\_SUCCESS; otherwise, it returns a Win32 error.

##### RemoveADLDSSPNs

1. procedure RemoveADLDSSPNs(szScriptBase: unicodestring): boolean

The RemoveADLDSSPNs procedure connects to an [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024) [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) and attempts to delete any [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) values registered for the [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) DC on the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) (in the external AD DS [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca)) that corresponds to the [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) that the AD LDS service is running as. Sections [2.2.3.2](#Section_41efc56e00074e88bafed7af61efd91f) and [2.2.4.2](#Section_894d09997d794e81a4077bcf6522b0a7) specify the SPN values removed by this procedure. If no such SPN values exist or the deletion is successful, RemoveADLDSSPNs returns TRUE; otherwise, it returns FALSE, indicating that a batch file was created in the folder specified by the *szScriptBase* parameter. This batch file contains commands that an administrator can run to clean up the SPNs.

**Note**  When the procedure fails to create a batch file for any reason, RemoveADLDSSPNs returns TRUE.

#### Server Behavior of the IDL\_DRSFinishDemotion Method

*Informative summary of behavior*: The [IDL\_DRSFinishDemotion](#Section_0bf530e81be04f48b8c2208031a8725f) method either performs one or more steps toward the complete removal of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) from an [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62), or it undoes the effects of the first phase of removal (performed by [IDL\_DRSInitDemotion](#Section_faca17da3f7f409298dbfd2ce7d98b8c)).[<16>](#Appendix_A_16" \o "Product behavior note 16)

1. ULONG
2. IDL\_DRSFinishDemotion(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_FINISH\_DEMOTIONREQ\* pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DRS\_MSG\_FINISH\_DEMOTIONREPLY\* pmsgOut
10. )
11. msgIn: DRS\_MSG\_FINISH\_DEMOTIONREQ\_V1
12. msgOut: DRS\_MSG\_FINISH\_DEMOTIONREPLY\_V1
13. ret: DWORD
14. res: boolean
15. ValidateDRSInput(hDrs, 27)
16. if dwInVersion ≠ 1 then
17. return ERROR\_INVALID\_PARAMETER
18. endif
19. if pmsgIn = null then
20. return ERROR\_INVALID\_PARAMETER
21. endif
22. msgIn := pmsgIn^.V1
23. if DS\_DEMOTE\_OPT\_FAIL\_ON\_UNKNOWN\_OP in msgIn.dwOperations then
24. /\* unknown operation bit is set \*/
25. return ERROR\_INVALID\_PARAMETER
26. endif
27. if DS\_DEMOTE\_UNREGISTER\_SPNS in msgIn.dwOperations
28. and msgIn.szScriptBase = null then
29. /\* szScriptBase must be specified when UNREGISTER\_SPN is
30. \* requested \*/
31. return ERROR\_INVALID\_PARAMETER
32. endif
33. if not IsMemberOfBuiltinAdminGroup() then
34. /\* only BA is allowed to demote an AD LDS service \*/
35. return ERROR\_DS\_DRA\_ACCESS\_DENIED
36. endif
37. pdwOutVersion^ := 1
38. msgOut.dwOperationDone := 0
39. msgOut.dwOpFailed := 0
40. msgOut.dwOpError := ERROR\_SUCCESS
41. if DS\_DEMOTE\_ROLLBACK\_DEMOTE in msgIn.dwOperations then
42. /\* Begin operations corresponding to dwOperations value of DS\_DEMOTE\_ROLLBACK\_DEMOTE \*/
43. /\* undo the effects of IDL\_DRSInitDemotion \*/
44. dc.fEnableUpdates := TRUE
45. msgOut.dwOperationDone :=
46. msgOut.dwOperationDone + {DS\_DEMOTE\_ROLLBACK\_DEMOTE}
47. msgOut.dwOpError := ERROR\_SUCCESS
48. /\* no other operations are allowed on rollback \*/
49. /\* End operations corresponding to dwOperations value of DS\_DEMOTE\_ROLLBACK\_DEMOTE \*/
50. else
51. if DS\_DEMOTE\_COMMIT\_DEMOTE in msgIn.dwOperations then
52. /\* Begin operations corresponding to dwOperations value of DS\_DEMOTE\_COMMIT\_DEMOTE \*/
54. After this call to IDL\_DRSFinishDemotion completes, the server must discontinue being a DC, which for AD LDS means stopping the MS-DRSR protocol, the MS-DSSP protocol, the LDAP protocol, and if they are already enabled also the MS-ADCAP protocol, the WS-Enumeration protocol, the WS-Transfer protocol, the MS-WSTIM protocol, the MS-WSDS protocol, and the MS-WSPELD protocol. In addition, the state model, constraints and processing rules, and so on, in MS-ADTS must also be stopped.
55. msgOut.dwOperationDone :=
56. msgOut.dwOperationDone + {DS\_DEMOTE\_COMMIT\_DEMOTE}
58. msgOut.dwOpError := ERROR\_SUCCESS
59. /\* End operations corresponding to dwOperations value of DS\_DEMOTE\_COMMIT\_DEMOTE \*/
60. endif
61. if DS\_DEMOTE\_DELETE\_CSMETA in msgIn.dwOperations then
62. /\* Begin operations corresponding to dwOperations value of DS\_DEMOTE\_DELETE\_CSMETA \*/
63. ret := RemoveADLDSServer()
64. if ret = ERROR\_SUCCESS then
65. msgOut.dwOperationDone :=
66. msgOut.dwOperationDone + {DS\_DEMOTE\_DELETE\_CSMETA}
67. else
68. msgOut.dwOpFailed =
69. msgOut.dwOpFailed + {DS\_DEMOTE\_DELETE\_CSMETA}
70. if msgOut.dwOpError = ERROR\_SUCCESS then
71. msgOut.dwOpError := ret
72. endif
73. endif
74. /\* End operations corresponding to dwOperations value of DS\_DEMOTE\_DELETE\_CSMETA \*/
75. endif
76. if DS\_DEMOTE\_UNREGISTER\_SCPS in msgIn.dwOperations then
77. /\* Begin operations corresponding to dwOperations value of DS\_DEMOTE\_UNREGISTER\_SCPS \*/
78. ret := RemoveADLDSSCP()
79. if ret = ERROR\_SUCCESS then
80. msgOut.dwOperationDone :=
81. msgOut.dwOperationDone + {DS\_DEMOTE\_UNREGISTER\_SCPS}
82. else
83. msgOut.dwOpFailed =
84. msgOut.dwOpFailed + {DS\_DEMOTE\_UNREGISTER\_SCPS}
85. if msgOut.dwOpError = ERROR\_SUCCESS then
86. msgOut.dwOpError := ret
87. endif
88. endif
89. /\* End operations corresponding to dwOperations value of DS\_DEMOTE\_UNREGISTER\_SCPS \*/
90. endif
91. if DS\_DEMOTE\_UNREGISTER\_SPNS in msgIn.dwOperations then
92. /\* Begin operations corresponding to dwOperations value of DS\_DEMOTE\_UNREGISTER\_SPNS \*/
93. res := RemoveADLDSSPNs(msgIn.szScriptBase)
94. if res = TRUE then
95. msgOut.dwOperationDone :=
96. msgOut.dwOperationDone + {DS\_DEMOTE\_UNREGISTER\_SPNS}
97. else
98. msgOut.dwOpFailed =
99. msgOut.dwOpFailed + {DS\_DEMOTE\_UNREGISTER\_SPNS}
100. endif
101. /\* End operations corresponding to dwOperations value of DS\_DEMOTE\_UNREGISTER\_SPNS \*/
102. endif
103. endif
104. pmsgOut^ := msgOut
105. pdwMsgOut^ := 1
106. return ERROR\_SUCCESS

### IDL\_DRSGetMemberships (Opnum 9)

The IDL\_DRSGetMemberships method retrieves [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) membership for an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

1. ULONG IDL\_DRSGetMemberships(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_REVMEMB\_REQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_REVMEMB\_REPLY\* pmsgOut
9. );

**hDrs:** [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** Version of the request message.

**pmsgIn:** Pointer to the request message.

**pdwOutVersion:** Pointer to the version of the response message.

**pmsgOut:** Pointer to the response message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_REVMEMB\_REQ

The DRS\_MSG\_REVMEMB\_REQ union defines the request messages sent to the [IDL\_DRSGetMemberships](#Section_d5ace4527cdd4d50bb6439b4c55180a2) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_REVMEMB\_REQ\_V1 V1;
6. } DRS\_MSG\_REVMEMB\_REQ;

**V1:**  Version 1 request. Currently only one version is defined.

##### DRS\_MSG\_REVMEMB\_REQ\_V1

The DRS\_MSG\_REVMEMB\_REQ\_V1 structure defines the request message sent to the [IDL\_DRSGetMemberships](#Section_d5ace4527cdd4d50bb6439b4c55180a2) method.

1. typedef struct {
2. [range(1,10000)] ULONG cDsNames;
3. [size\_is(cDsNames,)] DSNAME\*\* ppDsNames;
4. DWORD dwFlags;
5. [range(1,7)] REVERSE\_MEMBERSHIP\_OPERATION\_TYPE OperationType;
6. DSNAME\* pLimitingDomain;
7. } DRS\_MSG\_REVMEMB\_REQ\_V1;

**cDsNames:**  The count of items in the **ppDsNames** array.

**ppDsNames:**  The [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) whose reverse membership is being requested, plus the DSNames of [**groups**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) of the appropriate type(s) of which it is already known to be a member.

**dwFlags:**  Zero or more of the following bit flags, which are presented in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | X | X | X | A | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**A (DRS\_REVMEMB\_FLAG\_GET\_ATTRIBUTES, 0x00000001):** Query the [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that correspond to the group membership.

**OperationType:** The type of group membership evaluation to be performed.

**pLimitingDomain:** [**Domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) filter; resulting objects that are not from this domain are neither returned nor followed transitively.

##### REVERSE\_MEMBERSHIP\_OPERATION\_TYPE

The REVERSE\_MEMBERSHIP\_OPERATION\_TYPE enumeration defines the type of reverse membership evaluation.

1. typedef enum
2. {
3. RevMembGetGroupsForUser = 1,
4. RevMembGetAliasMembership,
5. RevMembGetAccountGroups,
6. RevMembGetResourceGroups,
7. RevMembGetUniversalGroups,
8. GroupMembersTransitive,
9. RevMembGlobalGroupsNonTransitive
10. } REVERSE\_MEMBERSHIP\_OPERATION\_TYPE;

**RevMembGetGroupsForUser:** Nontransitive membership in [**groups**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) that are confined to a given [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), excluding built-in groups and domain-local groups. See [[MS-ADSC]](%5bMS-ADSC%5d.pdf#Section_9abb5e97123d4da99557b353ab79b830) section 2.14.

**RevMembGetAliasMembership:** Nontransitive membership in domain-local groups that are confined to a given domain.

**RevMembGetAccountGroups:** Transitive membership in all account groups in a given domain, excluding built-in groups.

**RevMembGetResourceGroups:** Transitive membership in all domain-local groups in a given domain, excluding built-in groups.

**RevMembGetUniversalGroups:** Transitive membership in all [**universal groups**](#gt_f46053d6-0708-4094-ac63-57c1bcb73d32), excluding built-in groups.

**GroupMembersTransitive:** Transitive closure of members of a group based on the information present in the server's [**NC replicas**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210), including the primary group.

**RevMembGlobalGroupsNonTransitive:** Non-transitive membership in [**global groups**](#gt_2002f42a-84dd-4401-ac8b-8088af87eae6), excluding built-in groups.

##### DRS\_MSG\_REVMEMB\_REPLY

The DRS\_MSG\_REVMEMB\_REPLY union defines the response messages received from the [IDL\_DRSGetMemberships](#Section_d5ace4527cdd4d50bb6439b4c55180a2) method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_REVMEMB\_REPLY\_V1 V1;
6. } DRS\_MSG\_REVMEMB\_REPLY;

**V1:**  Version 1 reply.

##### DRS\_MSG\_REVMEMB\_REPLY\_V1

The DRS\_MSG\_REVMEMB\_REPLY\_V1 structure defines the response message received from the [IDL\_DRSGetMemberships](#Section_d5ace4527cdd4d50bb6439b4c55180a2) method.

1. typedef struct {
2. ULONG errCode;
3. [range(0,10000)] ULONG cDsNames;
4. [range(0,10000)] ULONG cSidHistory;
5. [size\_is(cDsNames,)] DSNAME\*\* ppDsNames;
6. [size\_is(cDsNames)] DWORD\* pAttributes;
7. [size\_is(cSidHistory,)] NT4SID\*\* ppSidHistory;
8. } DRS\_MSG\_REVMEMB\_REPLY\_V1;

**errCode:**  0 on success. On failure, this can be one of the following.

| Value | Meaning |
| --- | --- |
| STATUS\_INSUFFICIENT\_RESOURCES  0xC000009A | Insufficient system resources exist to complete the request. |
| STATUS\_TOO\_MANY\_CONTEXT\_IDS  0xC000015A | The number of [**groups**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) is greater than the number that can be returned to the caller. |

**cDsNames:**  Count of items in the **ppDsNames** array.

**cSidHistory:**  Count of items in the **ppSidHistory** array.

**ppDsNames:**  The filtered group membership. This list contains the combined membership for all the names specified in ppDsNames field of the input [DRS\_MSG\_REVMEMB\_REQ\_V1](#Section_bc96a03b579e44548db412067b6ca985) structure.

**pAttributes:**  Properties of the returned groups. Values are chosen from SE\_GROUP values.

**ppSidHistory:**  [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) history of the returned groups.

##### SE\_GROUP Values

[**Attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of a security [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac).

| Symbolic name | Value |
| --- | --- |
| SE\_GROUP\_MANDATORY | 0x00000001 |
| SE\_GROUP\_ENABLED\_BY\_DEFAULT | 0x00000002 |
| SE\_GROUP\_ENABLED | 0x00000004 |

#### Method-Specific Abstract Types and Procedures

##### Arc and ArcSet

1. type Arc = [initial: DSName, final: DSName]
2. type ArcSet = set of Arc

##### Closure

1. procedure Closure(
2. vSet: set of DSName,
3. aSet: ArcSet,
4. v: DSName): set of DSName

The Closure procedure returns the set of vertices that can be reached from vertex *v* in the directed graph that consists of vertex set *vSet* and arc set *aSet*. A vertex *u* can be reached from *v* if and only if there is a sequence *v*[0], *v*[1], ... , *v*[*k*], where *v*[0]=*v*, *v*[*k*]=*u*, and *v*[*i*] is in *vSet* and [initial:*v*[*i*-1], final:*v*[*i]*) is in *aSet* for *i*=1,2,...,*k*.

##### DomainOf

1. procedure DomainOf(o: DSName): DSName

The DomainOf procedure returns the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) to which the given DSName *o* belongs. It returns null upon failure.

##### GetDSNameOfEnterpriseRODCsGroup

1. procedure GetDSNameOfEnterpriseReadonlyDomainControllerGroup(): DSName

This procedure constructs a [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) s consisting of the [**domain SID**](#gt_c1d6ba4d-2302-43a5-acd2-02bfe19d0ade) of the root domain and the [**relative identifier (RID)**](#gt_df3d0b61-56cd-4dac-9402-982f1fedc41c) of the Enterprise Read-only Domain Controllers Group (as defined in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.1.6.14), and returns the **DSName** of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *o* for which *o*! **objectSid** = *s*. If no such object *o* exists, this procedure returns null.

##### GetDSNameFromPrimaryGroupId

1. procedure GetDSNameFromPrimaryGroupId(rid: Rid): DSName

This procedure constructs a [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) s consisting of the [**domain SID**](#gt_c1d6ba4d-2302-43a5-acd2-02bfe19d0ade) of the [**DC's**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) default [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) and the given relative identifier **rid**, and returns the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *o* for which *o*!objectSid = *s*. If no such object *o* exists, then this procedure will return null.

##### IsMatchedGroup

1. procedure IsMatchedGroup(
2. w: DSName,
3. op: REVERSE\_MEMBERSHIP\_OPERATION\_TYPE,
4. limitingDomain: DSName): boolean

*Informative summary of behavior*: The IsMatchedGroup procedure checks whether an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) should be included in the result for the specified [IDL\_DRSGetMemberships](#Section_d5ace4527cdd4d50bb6439b4c55180a2) operation.

1. limitToDomain, filteroutBuiltin, result: boolean
2. w: DSName
3. limitToDomain := (op ≠ RevMembGetUniversalGroups) and
4. (limitingDomain ≠ null)
5. filteroutBuiltin := (op ≠ RevMembGetAliasMembership)
6. result := (GROUP\_TYPE\_SECURITY\_ENABLED in w!groupType)
7. and ((not limitToDomain) or (limitingDomain = DomainOf(w)))
8. and ((not filteroutBuiltin) or (not IsBuiltinPrincipal(w.sid)))
9. and ((op ≠ RevMembGetGroupsForUser)
10. or (w!groupType ∩ {GROUP\_TYPE\_RESOURCE\_GROUP,
11. GROUP\_TYPE\_APP\_BASIC\_GROUP,
12. GROUP\_TYPE\_APP\_QUERY\_GROUP} = {}))
13. and ((op ≠ RevMembGetAliasMembership)
14. or (w!groupType ∩ {GROUP\_TYPE\_RESOURCE\_GROUP,
15. GROUP\_TYPE\_APP\_BASIC\_GROUP,
16. GROUP\_TYPE\_APP\_QUERY\_GROUP} ≠ {}))
17. and ((op ≠ RevMembGetAccountGroups)
18. or (GROUP\_TYPE\_ACCOUNT\_GROUP in w!groupType))
19. and ((op ≠ RevMembGetResouceGroups)
20. or (GROUP\_TYPE\_RESOURCE\_GROUP in w!groupType))
21. and ((op ≠ RevMembGetUniversalGroups)
22. or (GROUP\_TYPE\_UNIVERSAL\_GROUP in w!groupType))
23. and ((op ≠ RevMembGlobalGroupsNonTransitive)
24. or (GROUP\_TYPE\_ACCOUNT\_GROUP in w!groupType))
25. return result

##### Neighbors

1. procedure Neighbors(
2. vSet: set of DSName,
3. aSet: ArcSet,
4. v: DSName): set of DSName

The Neighbors procedure returns the set of vertices adjacent to vertex *v* in the directed graph that consists of vertex set *vSet* and arc set *aSet*. A vertex *u* is adjacent to *v* if *u* is in *vSet* and [initial:*v*, final:*u*] is in *aSet*. Note that because this is a directed graph, the fact that vertex *u* is adjacent to vertex *v* does not imply that vertex *v* is adjacent to vertex *u*.

#### Server Behavior of the IDL\_DRSGetMemberships Method

*Informative summary of behavior*: The [IDL\_DRSGetMemberships](#Section_d5ace4527cdd4d50bb6439b4c55180a2) method constructs a directed graph G(V,A). The vertex set of the graph includes all the [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the scope of the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) if the server is a [**GC**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169), or in the scope of the default [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) otherwise. The arc set of the graph includes all the tuples [initial: *u*,final: *v*] if *u* is a member of *v* and both *u* and *v* are in the scope. This graph represents the membership relation in the given scope.

For a GroupMembersTransitive request, a reversed graph of G is used because member relation is queried rather than membership. The reversed graph has the same vertex set as G, but the arcs in the arc set are in the opposite direction as those in A.

For other types of requests, a subgraph of G is used. The vertex set of this subgraph consists of only the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) values of interest for that particular request type, and the arc set is reduced to the arcs that link two vertices in the vertex set of the subgraph.

Starting from the graph, this method computes a set of objects for each DSName in the input parameters. The set could be either transitive closure of the object or the immediate neighbors of the object in the graph, depending on the type of request. The union of these sets is returned as the result.

1. ULONG
2. IDL\_DRSGetMemberships(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_REVMEMB\_REQ \*pmsgIn,
6. [out, ref] DWORD \*pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_REVMEMB\_REPLY \*pmsgOut)
9. msgIn: DRS\_MSG\_REVMEMB\_REQ\_V1
10. vSet, wSet, uSet: set of DSName
11. aSet, aSetR: ArcSet
12. u,v,w: DSName
13. op, i: integer
14. transitive: boolean
15. t: SID
16. ValidateDRSInput(hDrs, 9)
17. pdwOutVersion^ := 1
18. pmsgOut^.V1.errCode := 0
19. pmsgOut^.V1.cDsNames := 0
20. pmsgOut^.V1.cSidHistory := 0
21. pmsgOut^.V1.ppDsNames := null
22. pmsgOut^.V1.pAttributes := null
23. pmsgOut^.V1.ppSidHistory := null
24. msgIn := pmsgIn^.V1
25. if dwInVersion ≠ 1 then
26. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
27. endif
28. if not AccessCheckCAR(DefaultNC(), DS-Replication-Get-Changes) then
29. return ERROR\_DS\_DRA\_ACCESS\_DENIED
30. endif
31. op := msgIn.OperationType
32. if (op = RevMembGetUniversalGroups) and not IsGC() then
33. return ERROR\_DS\_GC\_REQUIRED
34. endif
35. /\* Construct a membership graph. \*/
36. /\* Vertices \*/
37. if IsGC() then
38. vSet := select all v from all where true
39. else
40. vSet := select all v from subtree DefaultNC() where true
41. Endif
42. /\* Edges \*/
43. aSet := {}
44. aSetR := {}
45. foreach v in vSet
46. foreach u in vSet
47. if (u in v!memberOf)
48. or (u = GetDSNameFromPrimaryGroupId(v!primaryGroupId)) then
49. aSet := aSet + {[initial: v, final: u]}
50. aSetR := aSetR + {[initial: u, final: v]}
51. endif
52. endfor
53. endfor
54. /\* Calculate GroupMembersTransitive. \*/
55. if op = GroupMembersTransitive then
56. wSet := {}
57. for i := 0 to msgIn.ppDsNames.cDsNames - 1
58. u := msgIn.ppDsNames[i]
59. if u in vSet then
60. wSet := wSet + (Closure(uSet, aSetR, u) - {u})
61. endif
62. endfor
63. foreach w in wSet
64. pmsgOut^.V1.ppDsNames[pmsgOut^.V1.cDsNames] := w
65. pmsgOut^.V1.cDsNames:= pmsgOut^.V1.cDsNames + 1
66. endfor
67. return 0
68. endif
69. /\* Calculate all other cases (where op ≠ GroupMembersInTransitive).\*/
70. transitive := op in {RevMembGetAccountGroups,
71. RevMembGetResourceGroups,
72. RevMembGetUniversalGroups}
73. /\* Get the initial result set from the graph. \*/
74. wSet := {}
75. for i := 0 to msgIn.ppDsNames.cDsNames - 1
76. u := msgIn.ppDsNames[i]
77. if u in vSet then
78. /\* Get the subgraph by applying the predicate IsMatchedGroup
79. \* on each element in the vertex set, plus u itself. \*/
80. uSet := {u} + select all v from vSet where
81. IsMatchedGroup(v, op, msgIn.pLimitingDomain^)
82. if transitive then
83. wSet := wSet + (Closure(uSet, aSet, u) - {u})
84. else
85. wSet := wSet + (Neighbors(uSet, aSet, u) - {u})
86. endif
87. if((u!userAccountControl & ADS\_UF\_WORKSTATION\_TRUST\_ACCOUNT =
88. ADS\_UF\_WORKSTATION\_TRUST\_ACCOUNT) and
89. (u!userAccountControl & ADS\_UF\_PARTIAL\_SECRETS\_ACCOUNT =
90. ADS\_UF\_PARTIAL\_SECRETS\_ACCOUNT))
91. wSet := wSet + GetDSNameOfEnterpriseRODCsGroup()
92. endif
93. endif
94. endfor
95. /\* Construct the result message. \*/
96. pmsgOut^.V1.cSidHistory := 0
97. pmsgOut^.V1.cDsNames := 0
98. foreach w in wSet
99. foreach t in w!sIDHistory
100. if not (t in pmsgOut^.V1.ppSidHistory) then
101. pmsgOut^.V1.ppSidHistory[pmsgOut^.V1.cSidHistory] := t
102. pmsgOut^.V1.cSidHistory := pmsgOut^.V1.cSidHistory + 1
103. endif
104. endfor
106. pmsgOut^.V1.ppDsNames[pmsgOut^.V1.cDsNames] := w
107. if (DRS\_REVMEMB\_FLAG\_GET\_ATTRIBUTES in msgIn.dwFlags) then
108. pmsgOut^.V1.pAttributes[pmsgOut^.V1.cDsNames] :=
109. {SE\_GROUP\_MANDATORY,SE\_GROUP\_ENABLED\_BY\_DEFAULT,
110. SE\_GROUP\_ENABLED}
111. else
112. pmsgOut^.V1.pAttributes[pmsgOut^.V1.cDsNames] := 0
113. endif
114. pmsgOut^.V1.cDsNames := pmsgOut^.V1.cDsNames + 1
115. endfor
116. return 0

### IDL\_DRSGetMemberships2 (Opnum 21)

The IDL\_DRSGetMemberships2 method retrieves [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) memberships for a sequence of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

1. ULONG IDL\_DRSGetMemberships2(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_GETMEMBERSHIPS2\_REQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_GETMEMBERSHIPS2\_REPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** Version of the request message.

**pmsgIn:** Pointer to the request message.

**pdwOutVersion:** Pointer to the version of the response message.

**pmsgOut:** Pointer to the response message.

**Return Values:** 0 if successful; otherwise, a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_GETMEMBERSHIPS2\_REQ

The DRS\_MSG\_GETMEMBERSHIPS2\_REQ union defines request messages sent to the [IDL\_DRSGetMemberships2](#Section_d4e67cc32ee14b2b8055cebefc556252) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_GETMEMBERSHIPS2\_REQ\_V1 V1;
6. } DRS\_MSG\_GETMEMBERSHIPS2\_REQ;

**V1:**  Version 1 request.

##### DRS\_MSG\_GETMEMBERSHIPS2\_REQ\_V1

The DRS\_MSG\_GETMEMBERSHIPS2\_REQ\_V1 structure defines the request message sent to the [IDL\_DRSGetMemberships2](#Section_d4e67cc32ee14b2b8055cebefc556252) method.

1. typedef struct {
2. [range(1,10000)] ULONG Count;
3. [size\_is(Count)] DRS\_MSG\_REVMEMB\_REQ\_V1\* Requests;
4. } DRS\_MSG\_GETMEMBERSHIPS2\_REQ\_V1;

**Count:**  Count of items in the **Requests** array.

**Requests:**  Sequence of reverse membership requests.

##### DRS\_MSG\_GETMEMBERSHIPS2\_REPLY

The DRS\_MSG\_GETMEMBERSHIPS2\_REPLY union defines response messages received from the [IDL\_DRSGetMemberships2](#Section_d4e67cc32ee14b2b8055cebefc556252) method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_GETMEMBERSHIPS2\_REPLY\_V1 V1;
6. } DRS\_MSG\_GETMEMBERSHIPS2\_REPLY;

**V1:**  Version 1 response.

##### DRS\_MSG\_GETMEMBERSHIPS2\_REPLY\_V1

The DRS\_MSG\_GETMEMBERSHIPS2\_REPLY\_V1 structure defines the response message received from the [IDL\_DRSGetMemberships2](#Section_d4e67cc32ee14b2b8055cebefc556252) method.

1. typedef struct {
2. [range(0,10000)] ULONG Count;
3. [size\_is(Count)] DRS\_MSG\_REVMEMB\_REPLY\_V1\* Replies;
4. } DRS\_MSG\_GETMEMBERSHIPS2\_REPLY\_V1;

**Count:**  Count of items in the **Replies** array.

**Replies:**  Sequence of reverse membership replies, in the same order as the **Requests** field of the request message.

#### Server Behavior of the IDL\_DRSGetMemberships2 Method

*Informative summary of behavior*: The [IDL\_DRSGetMemberships2](#Section_d4e67cc32ee14b2b8055cebefc556252) method is merely a way to execute a series of [IDL\_DRSGetMemberships](#Section_d5ace4527cdd4d50bb6439b4c55180a2) [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) calls via a single RPC request.

1. ULONG
2. IDL\_DRSGetMemberships2(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_GETMEMBERSHIPS2\_REQ \*pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DRS\_MSG\_GETMEMBERSHIPS2\_REPLY \*pmsgOut)
10. error, i: ULONG
11. dummyVersion: DWORD
12. ValidateDRSInput(hDrs, 21)
13. pdwOutVersion^ := 1
14. pMsgOut^.V1.Count := 0
15. pMsgOut^.V1.Replies := null
16. if dwInVersion ≠ 1 then
17. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
18. endif
19. pmsgOut^.V1.Count := pmsgIn^.V1.Count
20. for i := 0 to pmsgIn^.V1.Count - 1
21. /\* Call IDL\_DRSGetMemberships as a local procedure. \*/
22. error := IDL\_DRSGetMemberships(null, 1, ADR(pmsgIn^.V1.Request[i]),
23. ADR(dummyVersion), ADR(pmsgOut^.V1.Replies[i]))
24. if error ≠ 0 then
25. return error
26. endif
27. endfor
28. return 0

### IDL\_DRSGetNCChanges (Opnum 3)

The IDL\_DRSGetNCChanges method replicates [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) from an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) on the server.

1. ULONG IDL\_DRSGetNCChanges(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_GETCHGREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_GETCHGREPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** Version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Overview

A client [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) sends an [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) request (*msgIn*, of a type in the union [DRS\_MSG\_GETCHGREQ](#Section_96affbe17d93453eac759f41c0c94b3b)) to a server to replicate [**directory objects**](#gt_407dbc2c-3140-4e31-9085-0087e2d3bab2) in a given [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) from the server [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) to the client NC replica.

The response (*msgOut*, of a type in the union [DRS\_MSG\_GETCHGREPLY](#Section_65a5cb42c25f4378b06ef87341b21f93)) contains a set of [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) that the client is to apply to its NC replica. Commonly, this set of updates is too large to send in a single response; in this case, multiple IDL\_DRSGetNCChanges requests and responses must be sent before the server sends a response that indicates no additional updates are available.

This sequence of requests and responses is called a [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16), or "cycle". A client DC can request an action on a [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f) (for example, a change in the [**FSMO role owner**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b)) by using a special replication cycle called an [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6).

##### Cycle Start and Finish

There are five types of [**cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16) starts:

1. The client explicitly signals the start of a special single-response cycle when it requests an [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6). Such cycles always consist of a single response which sets msgOut.fMoreData = false.
2. The client explicitly signals the start of a cycle by sending msgIn.uuidInvocIdSrc = 0 or msgIn.usnvecFrom = 0.
3. The client sends values of msgIn.uuidInvocIdSrc and msgIn.usnvecFrom that were returned by the server as msgOut.uuidInvocIdSrc and msgOut.usnvecTo in the final message of some other cycle.
4. The server detects either that msgIn.uuidInvocIdSrc ≠ [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).invocationId or that msgIn.uuidInvocIdSrc and msgIn.usnvecFrom were not returned by the server in the final message of some other cycle.
5. The server implementation MAY[<17>](#Appendix_A_17" \o "Product behavior note 17) declare the supplied values of msgIn.uuidInvocIdSrc and msgIn.usnvecFrom as too stale to use.

If the server starts a new cycle based on items 4 or 5, the server ignores msgIn.usnvecFrom, treating it as though it were zero.

The fields msgOut.usnvecTo and msgIn.usnvecFrom have the same type, [USN\_VECTOR](#Section_595d11b86ca74a61bd563e6a2b99b76b). The internal format of USN\_VECTOR is determined entirely by the server implementation and is subject only to the requirement that msgIn.usnvecFrom = 0 represents the start of a cycle. The server MAY[<18>](#Appendix_A_18" \o "Product behavior note 18) use USN\_VECTOR to encode the start of a cycle.

Any server response message with msgOut.fMoreData = false is the final response in a cycle.

##### Cycle Goal

For any [**cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16) that is not an [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6), the goal of the server is to send [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) such that at the conclusion of the cycle, the client [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) contains all updates that were present in the server NC replica at the start of the cycle. More concretely, if *cycleStartUtd* is the server's *msgIn.pNC^!*replUpToDateVector on receipt of the first request in a cycle where *msgIn.ulExtendedOp* = 0, then the final response in the cycle MUST contain *msgOut.pUpToDateVecSrc* such that HasUpdateKnowledge(*msgOut.pUpToDateVecSrc^*, *cycleStartUtd*) = true:

1. procedure HasUpdateKnowledge(
2. utd1: UPTODATE\_VECTOR\_V2\_EXT,
3. utd2: UPTODATE\_VECTOR\_V2\_EXT): boolean
4. begin
5. i: integer
6. j: integer
7. /\* Return true if and only if utd1 asserts the presence of all
8. \* updates asserted by utd2. \*/
9. for i := 0 to utd2.cNumCursors - 1
10. j := select one k from [0 .. utd1.cNumCursors - 1] where
11. utd1.rgCursors[k].uuidDsa = utd2.rgCursors[i].uuidDsa
12. if j = null or utd1.rgCursors[j].usnHighPropUpdate <
13. utd2.rgCursors[i].usnHighPropUpdate then
14. return false
15. endif
16. endfor
17. return true
18. end HasUpdateKnowledge

The server MAY[<19>](#Appendix_A_19" \o "Product behavior note 19) advance the cycle goal on each request such that it includes updates that the server has applied since the first request in the cycle.

The cycle goal includes a cursor *c* for the server [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) such that:

* *c.uuidDsa* is the value of the invocationId [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of the server's nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).
* *c.usnHighPropUpdate* is the highest [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) such that the server can assert that, including the updates in this response, the client has applied any update with [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) s where *s.uuidOriginating* = *c.uuidDsa* and *s.usnOriginating* ≤ *c.usnHighPropUpdate*. If the server has originated no updates in the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942), it MAY[<20>](#Appendix_A_20" \o "Product behavior note 20) set *c.usnHighPropUpdate* to 0.
* *c.timeLastSyncSuccess* is the time at which the server sends the final response.

##### Extended Operations

The [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6) specified by *msgIn.ulExtendedOp* is one of the following:

* Request Role (EXOP\_FSMO\_REQ\_ROLE, EXOP\_FSMO\_REQ\_PDC, EXOP\_FSMO\_RID\_REQ\_ROLE): Changes the [**FSMO role owner**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b) from the server to the client [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), and then adds all changed [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) and [**link values**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) in the [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f) to the response, including but not limited to the FSMO role owner change.[<21>](#Appendix_A_21" \o "Product behavior note 21)
* Abandon Role (EXOP\_FSMO\_ABANDON\_ROLE): Performs a chained request to the current FSMO role owner to make the server DC the FSMO role owner. This request is sent to help avoid entering a state in which no DC considers itself the owner of the role.[<22>](#Appendix_A_22" \o "Product behavior note 22)
* Allocate [**RIDs**](#gt_df3d0b61-56cd-4dac-9402-982f1fedc41c) (EXOP\_FSMO\_REQ\_RID\_ALLOC): Allocates a new block of RIDs to the client DC.[<23>](#Appendix_A_23" \o "Product behavior note 23)
* Replicate Single Object (EXOP\_REPL\_OBJ): Adds any changes to the specified object to the response.[<24>](#Appendix_A_24" \o "Product behavior note 24)
* Replicate Single Object including Secret Data (EXOP\_REPL\_SECRETS): Adds any changes to the specified object to the response. In addition, it also adds the secret [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) values of the specified object to the response, regardless of whether they have recent changes. See the IsSecretAttribute procedure in section [4.1.10.3.11](#Section_294168d981bf461b91d795bd8a985737) for a list of these attributes.[<25>](#Appendix_A_25" \o "Product behavior note 25)

#### Method-Specific Concrete Types

##### DRS\_MSG\_GETCHGREQ

The DRS\_MSG\_GETCHGREQ union defines request messages that are sent to the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method. There are no V1, V2, V3, V6, or V9 messages.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(4)]
5. DRS\_MSG\_GETCHGREQ\_V4 V4;
6. [case(5)]
7. DRS\_MSG\_GETCHGREQ\_V5 V5;
8. [case(7)]
9. DRS\_MSG\_GETCHGREQ\_V7 V7;
10. [case(8)]
11. DRS\_MSG\_GETCHGREQ\_V8 V8;
12. [case(10)]
13. DRS\_MSG\_GETCHGREQ\_V10 V10;
14. [case(11)]
15. DRS\_MSG\_GETCHGREQ\_V11 V11;
16. } DRS\_MSG\_GETCHGREQ;

**V4:**  Version 4 request (Windows 2000 operating system SMTP [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) [[MS-SRPL]](%5bMS-SRPL%5d.pdf#Section_ec69eea50d5e428ab5bc66732aaeb866)).

**V5:**  Version 5 request (Windows 2000 [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) replication).

**V7:**  Version 7 request (Windows Server 2003 SMTP replication [MS-SRPL]).

**V8:**  Version 8 request (Windows Server 2003 RPC replication).

**V10:**  Version 10 request (Windows Server 2008 R2 operating system RPC replication).

**V11**:  Version 11 request (Windows Server v1803 operating system RPC replication).

##### DRS\_MSG\_GETCHGREQ\_V3

The DRS\_MSG\_GETCHGREQ\_V3 structure defines a portion of the request message that is sent to the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method as part of SMTP [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) ([[MS-SRPL]](%5bMS-SRPL%5d.pdf#Section_ec69eea50d5e428ab5bc66732aaeb866)). This is not a complete request message; it is embedded in [DRS\_MSG\_GETCHGREQ\_V4](#Section_9db4db218ccd4c8186626e2baff8426c) and [DRS\_MSG\_GETCHGREQ\_V7](#Section_5ef4f597a3974f6fa98b7a034247d886).[<26>](#Appendix_A_26" \o "Product behavior note 26)

1. typedef struct {
2. UUID uuidDsaObjDest;
3. UUID uuidInvocIdSrc;
4. [ref] DSNAME\* pNC;
5. USN\_VECTOR usnvecFrom;
6. [unique] UPTODATE\_VECTOR\_V1\_EXT\* pUpToDateVecDestV1;
7. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT\* pPartialAttrVecDestV1;
8. SCHEMA\_PREFIX\_TABLE PrefixTableDest;
9. ULONG ulFlags;
10. ULONG cMaxObjects;
11. ULONG cMaxBytes;
12. ULONG ulExtendedOp;
13. } DRS\_MSG\_GETCHGREQ\_V3;

**uuidDsaObjDest:**  [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the client [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**uuidInvocIdSrc:**  [**Invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the server DC.

**pNC:**  [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) to replicate or the [**FSMO role object**](#gt_6ea17c3e-787a-40e3-a62f-0313dcdc16b7) for an [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6).

**usnvecFrom:**  Data that is used to correlate calls to IDL\_DRSGetNCChanges.

**pUpToDateVecDestV1:**  [**Stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) filter describing [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) that the client has already applied.

**pPartialAttrVecDestV1:**  A set of one or more [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) whose values are to be replicated to the client's partial replica.

**PrefixTableDest:**  [**Prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) with which to convert the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values in pPartialAttrVecDestV1 to [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

**ulFlags:**  A [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) bit field.

**cMaxObjects:**  An approximate cap on the number of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to include in the reply.

**cMaxBytes:**  An approximate cap on the number of bytes to include in the reply.

**ulExtendedOp:**  0 or an EXOP\_REQ code (section [4.1.10.2.22](#Section_05de65eeaf0d46d1a9c84f0f856031cb)).

##### DRS\_MSG\_GETCHGREQ\_V4

The DRS\_MSG\_GETCHGREQ\_V4 structure defines the request message sent to the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method. This message version is a superset of [DRS\_MSG\_GETCHGREQ\_V3](#Section_6a2a056cac7f47d09e6d9023a4e5947c).[<27>](#Appendix_A_27" \o "Product behavior note 27)

1. typedef struct {
2. UUID uuidTransportObj;
3. [ref] MTX\_ADDR\* pmtxReturnAddress;
4. DRS\_MSG\_GETCHGREQ\_V3 V3;
5. } DRS\_MSG\_GETCHGREQ\_V4;

**uuidTransportObj:**  The objectGUID of the interSiteTransport [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that identifies the transport by which to send the reply.

**pmtxReturnAddress:**  The transport-specific address to which to send the reply.

**V3:**  Version 3 request.

##### DRS\_MSG\_GETCHGREQ\_V5

The DRS\_MSG\_GETCHGREQ\_V5 structure defines the request message sent to the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method.

1. typedef struct {
2. UUID uuidDsaObjDest;
3. UUID uuidInvocIdSrc;
4. [ref] DSNAME\* pNC;
5. USN\_VECTOR usnvecFrom;
6. [unique] UPTODATE\_VECTOR\_V1\_EXT\* pUpToDateVecDestV1;
7. ULONG ulFlags;
8. ULONG cMaxObjects;
9. ULONG cMaxBytes;
10. ULONG ulExtendedOp;
11. ULARGE\_INTEGER liFsmoInfo;
12. } DRS\_MSG\_GETCHGREQ\_V5;

**uuidDsaObjDest:**  [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the client [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**uuidInvocIdSrc**:  [**Invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the server DC.

**pNC:**  [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) to replicate or the [**FSMO role object**](#gt_6ea17c3e-787a-40e3-a62f-0313dcdc16b7) for an [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6).

**usnvecFrom:**  Data used to correlate calls to IDL\_DRSGetNCChanges.

**pUpToDateVecDestV1:**  [**Stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) filter that describes [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the client has already applied.

**ulFlags:**  [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) bit field.

**cMaxObjects:**  Approximate cap on the number of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to include in the reply.

**cMaxBytes:**  Approximate cap on the number of bytes to include in the reply.

**ulExtendedOp:**  0 or an extended operation request code (section [4.1.10.2.22](#Section_05de65eeaf0d46d1a9c84f0f856031cb)).

**liFsmoInfo:**  0 or a value specific to the requested extended operation.

##### DRS\_MSG\_GETCHGREQ\_V7

The DRS\_MSG\_GETCHGREQ\_V7 structure defines the request message sent to the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method. This message version is a superset of [DRS\_MSG\_GETCHGREQ\_V4](#Section_9db4db218ccd4c8186626e2baff8426c).[<28>](#Appendix_A_28" \o "Product behavior note 28)

1. typedef struct {
2. UUID uuidTransportObj;
3. [ref] MTX\_ADDR\* pmtxReturnAddress;
4. DRS\_MSG\_GETCHGREQ\_V3 V3;
5. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT\* pPartialAttrSet;
6. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT\* pPartialAttrSetEx;
7. SCHEMA\_PREFIX\_TABLE PrefixTableDest;
8. } DRS\_MSG\_GETCHGREQ\_V7;

**uuidTransportObj:**  The objectGUID of the interSiteTransport [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that identifies the transport by which to send the reply.

**pmtxReturnAddress:**  Transport-specific address to which to send the reply.

**V3:**  Version 3 request.

**pPartialAttrSet:**  A set of one or more [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) whose values are to be replicated to the client's partial replica, or null if the client has a full [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac).

**pPartialAttrSetEx:**  A set of one or more attributes whose values are to be added to the client's existing partial replica, or null.

**PrefixTableDest:**  [**Prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) with which to convert the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values in **pPartialAttrSet** and **pPartialAttrSetEx** to [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

##### DRS\_MSG\_GETCHGREQ\_V8

The DRS\_MSG\_GETCHGREQ\_V8 structure defines the request message sent to the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method. This message version is a superset of [DRS\_MSG\_GETCHGREQ\_V5](#Section_fd24b73c7b8143af8c7765bc2e3181b7).

1. typedef struct {
2. UUID uuidDsaObjDest;
3. UUID uuidInvocIdSrc;
4. [ref] DSNAME\* pNC;
5. USN\_VECTOR usnvecFrom;
6. [unique] UPTODATE\_VECTOR\_V1\_EXT\* pUpToDateVecDest;
7. ULONG ulFlags;
8. ULONG cMaxObjects;
9. ULONG cMaxBytes;
10. ULONG ulExtendedOp;
11. ULARGE\_INTEGER liFsmoInfo;
12. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT\* pPartialAttrSet;
13. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT\* pPartialAttrSetEx;
14. SCHEMA\_PREFIX\_TABLE PrefixTableDest;
15. } DRS\_MSG\_GETCHGREQ\_V8;

**uuidDsaObjDest:**  [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the client [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**uuidInvocIdSrc:**  [**Invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the server DC.

**pNC:**  [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) to replicate or the [**FSMO role object**](#gt_6ea17c3e-787a-40e3-a62f-0313dcdc16b7) for an [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6).

**usnvecFrom:**  Data used to correlate calls to IDL\_DRSGetNCChanges.

**pUpToDateVecDest:**  [**Stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) filter describing [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the client has already applied.

**ulFlags:**  A [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) bit field.

**cMaxObjects:**  Approximate cap on the number of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to include in the reply.

**cMaxBytes:**  Approximate cap on the number of bytes to include in the reply.

**ulExtendedOp:**  0 or an extended operation request code (section [4.1.10.2.22](#Section_05de65eeaf0d46d1a9c84f0f856031cb)).

**liFsmoInfo:**  0 or a value specific to the requested extended operation.

**pPartialAttrSet:**  A set of one or more [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) whose values are to be replicated to the client's partial replica, or null if the client has a full replica.

**pPartialAttrSetEx:**  A set of one or more attributes whose values are to be added to the client's existing partial replica, or null.

**PrefixTableDest:**  [**Prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) with which to convert the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values in **pPartialAttrSet** and **pPartialAttrSetEx** to [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

##### DRS\_MSG\_GETCHGREQ\_V10

The DRS\_MSG\_GETCHGREQ\_V10 structure defines the request message sent to the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method. This message version is a superset of [DRS\_MSG\_GETCHGREQ\_V8](#Section_4304bb4ae9b54c8a8731df4d6f9ab567).

1. typedef struct {
2. UUID uuidDsaObjDest;
3. UUID uuidInvocIdSrc;
4. [ref] DSNAME\* pNC;
5. USN\_VECTOR usnvecFrom;
6. [unique] UPTODATE\_VECTOR\_V1\_EXT\* pUpToDateVecDest;
7. ULONG ulFlags;
8. ULONG cMaxObjects;
9. ULONG cMaxBytes;
10. ULONG ulExtendedOp;
11. ULARGE\_INTEGER liFsmoInfo;
12. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT\* pPartialAttrSet;
13. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT\* pPartialAttrSetEx;
14. SCHEMA\_PREFIX\_TABLE PrefixTableDest;
15. ULONG ulMoreFlags;
16. } DRS\_MSG\_GETCHGREQ\_V10;

**uuidDsaObjDest:**  [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the client [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**uuidInvocIdSrc:**  [**Invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the server DC.

**pNC:**  [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) to replicate or the [**FSMO role object**](#gt_6ea17c3e-787a-40e3-a62f-0313dcdc16b7) for an [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6).

**usnvecFrom:**  Data used to correlate calls to IDL\_DRSGetNCChanges.

**pUpToDateVecDest:**  [**Stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) filter describing [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the client has already applied.

**ulFlags:**  A [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) bit field.

**cMaxObjects:**  Approximate cap on the number of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to include in the reply.

**cMaxBytes:**  Approximate cap on the number of bytes to include in the reply.

**ulExtendedOp:**  0 or an extended operation request code (section [4.1.10.2.22](#Section_05de65eeaf0d46d1a9c84f0f856031cb)).

**liFsmoInfo:**  0 or a value specific to the requested extended operation.

**pPartialAttrSet:**  A set of one or more [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) whose values are to be replicated to the client's partial replica, or null if the client has a full replica.

**pPartialAttrSetEx:**  A set of one or more attributes whose values are to be added to the client's existing partial replica, or null.

**PrefixTableDest:**  [**Prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) with which to convert the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values in **pPartialAttrSet** and **pPartialAttrSetEx** to [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

**ulMoreFlags:**  A [DRS\_MORE\_GETCHGREQ\_OPTIONS](#Section_d16a3483d4e34f7eb4daed8cded8d970) bit field.

##### DRS\_MSG\_GETCHGREQ\_V11

The DRS\_MSG\_GETCHGREQ\_V11 structure defines the request message sent to the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method. This message version is a superset of [DRS\_MSG\_GETCHGREQ\_V10](#Section_92b1b77d205846e09e8c6664b96a0cf9).

1. typedef struct {
2. UUID uuidDsaObjDest;
3. UUID uuidInvocIdSrc;
4. [ref] DSNAME\* pNC;
5. USN\_VECTOR usnvecFrom;
6. [unique] UPTODATE\_VECTOR\_V1\_EXT\* pUpToDateVecDest;
7. ULONG ulFlags;
8. ULONG cMaxObjects;
9. ULONG cMaxBytes;
10. ULONG ulExtendedOp;
11. ULARGE\_INTEGER liFsmoInfo;
12. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT\* pPartialAttrSet;
13. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT\* pPartialAttrSetEx;
14. SCHEMA\_PREFIX\_TABLE PrefixTableDest;
15. ULONG ulMoreFlags;
16. GUID correlationID;
17. [unique] VAR\_SIZE\_BUFFER\_WITH\_VERSION\* pReservedBuffer;
18. } DRS\_MSG\_GETCHGREQ\_V11;

**uuidDsaObjDest:**  [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the client [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**uuidInvocIdSrc:**  [**Invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the server DC.

**pNC:**  [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) to replicate or the [**FSMO role object**](#gt_6ea17c3e-787a-40e3-a62f-0313dcdc16b7) for an [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6).

**usnvecFrom:**  Data used to correlate calls to IDL\_DRSGetNCChanges.

**pUpToDateVecDest:**  [**Stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) filter describing [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the client has already applied.

**ulFlags:**  A [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) bit field.

**cMaxObjects:**  Approximate cap on the number of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to include in the reply.

**cMaxBytes:**  Approximate cap on the number of bytes to include in the reply.

**ulExtendedOp:**  0 or an extended operation request code (section [4.1.10.2.22](#Section_05de65eeaf0d46d1a9c84f0f856031cb)).

**liFsmoInfo:**  0 or a value specific to the requested extended operation.

**pPartialAttrSet:**  A set of one or more [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) whose values are to be replicated to the client's partial replica, or null if the client has a full replica.

**pPartialAttrSetEx:**  A set of one or more attributes whose values are to be added to the client's existing partial replica, or null.

**PrefixTableDest:**  [**Prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) with which to convert the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values in **pPartialAttrSet** and **pPartialAttrSetEx** to [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

**ulMoreFlags:**  A [DRS\_MORE\_GETCHGREQ\_OPTIONS](#Section_d16a3483d4e34f7eb4daed8cded8d970) bit field.

**correlationID**: An identifier for the operation that the DC can use for implementation-defined troubleshooting. There are no normative constraints on this value, nor does the value figure in any normative processing rules.

**pReservedBuffer**: A pointer to a VAR\_SIZE\_BUFFER\_WITH\_VERSION structure (section [5.219](#Section_589574c1eaa1456fac53de597b2cff6b)). MUST be a null pointer.

##### DRS\_MSG\_GETCHGREPLY

The DRS\_MSG\_GETCHGREPLY union defines the response messages received from the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method. There are no V3, V4, V5, or V8 messages.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_GETCHGREPLY\_V1 V1;
6. [case(2)]
7. DRS\_MSG\_GETCHGREPLY\_V2 V2;
8. [case(6)]
9. DRS\_MSG\_GETCHGREPLY\_V6 V6;
10. [case(7)]
11. DRS\_MSG\_GETCHGREPLY\_V7 V7;
12. [case(9)]
13. DRS\_MSG\_GETCHGREPLY\_V9 V9;
14. } DRS\_MSG\_GETCHGREPLY;

**V1:**  Version 1 response (Windows 2000).

**V2:**  Version 2 response (compressed V1).

**V6:**  Version 6 response (Windows Server 2003).

**V7:**  Version 7 response (compressed V6 or V9).

**V9**: Version 9 response (V6 with additional link-value metadata).

##### DRS\_MSG\_GETCHGREPLY\_V1

The DRS\_MSG\_GETCHGREPLY\_V1 structure defines the response message received from the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method.

1. typedef struct {
2. UUID uuidDsaObjSrc;
3. UUID uuidInvocIdSrc;
4. [unique] DSNAME\* pNC;
5. USN\_VECTOR usnvecFrom;
6. USN\_VECTOR usnvecTo;
7. [unique] UPTODATE\_VECTOR\_V1\_EXT\* pUpToDateVecSrcV1;
8. SCHEMA\_PREFIX\_TABLE PrefixTableSrc;
9. ULONG ulExtendedRet;
10. ULONG cNumObjects;
11. ULONG cNumBytes;
12. [unique] REPLENTINFLIST\* pObjects;
13. BOOL fMoreData;
14. } DRS\_MSG\_GETCHGREPLY\_V1;

**uuidDsaObjSrc:**  [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the server [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**uuidInvocIdSrc:**  [**Invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the server DC.

**pNC:**  The [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) to replicate or the [**FSMO role object**](#gt_6ea17c3e-787a-40e3-a62f-0313dcdc16b7) for an [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6).

**usnvecFrom:**  Data used to correlate calls to IDL\_DRSGetNCChanges.

**usnvecTo:**  Data used to correlate calls to IDL\_DRSGetNCChanges.

**pUpToDateVecSrcV1:**  [**Stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) filter that describes [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the server has already applied.

**PrefixTableSrc:**  Table for translating [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values in the response to [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

**ulExtendedRet:**  0 or an EXOP\_ERR code (section [4.1.10.2.21](#Section_40f60821c7d34d0785563398b1335a7f)).

**cNumObjects:**  Count of items in the **pObjects** linked list.

**cNumBytes:**  Size in bytes of items in or referenced by elements in the **pObjects** linked list.

**pObjects:**  Linked list of [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) updates that the client applies to its [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

**fMoreData:**  False if and only if this is the last response in a [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16).

##### DRS\_MSG\_GETCHGREPLY\_V2

The DRS\_MSG\_GETCHGREPLY\_V2 structure defines the compressed [DRS\_MSG\_GETCHGREPLY\_V1](#Section_bd70a9c3c1d348cf9c24503a5567d09c) message received from the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method.

1. typedef struct {
2. DRS\_COMPRESSED\_BLOB CompressedV1;
3. } DRS\_MSG\_GETCHGREPLY\_V2;

**CompressedV1:**  Compressed DRS\_MSG\_GETCHGREPLY\_V1 response.

##### DRS\_MSG\_GETCHGREPLY\_V6

The DRS\_MSG\_GETCHGREPLY\_V6 structure defines the response message received from the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method. This message version is a superset of [DRS\_MSG\_GETCHGREPLY\_V1](#Section_bd70a9c3c1d348cf9c24503a5567d09c).

1. typedef struct {
2. UUID uuidDsaObjSrc;
3. UUID uuidInvocIdSrc;
4. [unique] DSNAME\* pNC;
5. USN\_VECTOR usnvecFrom;
6. USN\_VECTOR usnvecTo;
7. [unique] UPTODATE\_VECTOR\_V2\_EXT\* pUpToDateVecSrc;
8. SCHEMA\_PREFIX\_TABLE PrefixTableSrc;
9. ULONG ulExtendedRet;
10. ULONG cNumObjects;
11. ULONG cNumBytes;
12. [unique] REPLENTINFLIST\* pObjects;
13. BOOL fMoreData;
14. ULONG cNumNcSizeObjects;
15. ULONG cNumNcSizeValues;
16. [range(0,1048576)] DWORD cNumValues;
17. [size\_is(cNumValues)] REPLVALINF\_V1\* rgValues;
18. DWORD dwDRSError;
19. } DRS\_MSG\_GETCHGREPLY\_V6;

**uuidDsaObjSrc:**  [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the server [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**uuidInvocIdSrc:**  [**Invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the server DC.

**pNC:**  The [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) to replicate or the [**FSMO role object**](#gt_6ea17c3e-787a-40e3-a62f-0313dcdc16b7) for an [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6).

**usnvecFrom:**  Data used to correlate calls to IDL\_DRSGetNCChanges.

**usnvecTo:**  Data used to correlate calls to IDL\_DRSGetNCChanges.

**pUpToDateVecSrc:**  [**Stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) filter that describes [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the server has already applied.

**PrefixTableSrc:**  Table for translating [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values in the response to [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

**ulExtendedRet:**  0 or an extended operation error code (section [4.1.10.2.21](#Section_40f60821c7d34d0785563398b1335a7f)).

**cNumObjects:**  Count of items in the **pObjects** linked list.

**cNumBytes:**  Size in bytes of items in or referenced by elements in the **pObjects** linked list.

**pObjects:**  Linked list of [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) updates that the client applies to its [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

**fMoreData:**  False if and only if this is the last response in a [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16).

**cNumNcSizeObjects:**  Estimated number of objects in the server's NC replica.

**cNumNcSizeValues:**  Estimated number of [**link values**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) in the server's NC replica.

**cNumValues:**  Count of items in the **rgValues** array.

**rgValues:**  Link value updates for the client to apply to its NC replica.

**dwDRSError:**  0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

##### DRS\_MSG\_GETCHGREPLY\_V7

The DRS\_MSG\_GETCHGREPLY\_V7 structure defines a compressed [DRS\_MSG\_GETCHGREPLY\_V6](#Section_1317a6545dd645ffaf73919cbc7fbb45) or [**DRS\_MSG\_GETCHGREPLY\_V9**](#Section_b9564a194500444ba99b0da1b08cdb6f) message received from the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method.

1. typedef struct {
2. DWORD dwCompressedVersion;
3. DRS\_COMP\_ALG\_TYPE CompressionAlg;
4. DRS\_COMPRESSED\_BLOB CompressedAny;
5. } DRS\_MSG\_GETCHGREPLY\_V7;

**dwCompressedVersion:**  Version of the response in **CompressedAny**; MUST be set to 6 or 9.

**CompressionAlg:**  Algorithm used to compress the response.

**CompressedAny:**  Compressed DRS\_MSG\_GETCHGREPLY\_V6 or DRS\_MSG\_GETCHGREPLY\_V9 response.

##### DRS\_MSG\_GETCHGREPLY\_V9

The **DRS\_MSG\_GETCHGREPLY\_V9** structure defines the response message received from the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method. This message version contains all the same elements as a [DRS\_MSG\_GETCHGREPLY\_V6](#Section_1317a6545dd645ffaf73919cbc7fbb45) structure except that the data type of **rgValues** is changed from REPLVALINF\_V1\* to REPLVALINF\_V3\*. The data in a [REPLVALINF\_V3](#Section_9c15369bb7d2437ab73d66a92c367795) structure is a superset of the data in a [**REPLVALINF\_V1**](#Section_22946fbf170e4ab482c7dabdfd97bf5a) structure. Therefore, the data in the **DRS\_MSG\_GETCHGREPLY\_V9** structure is a superset of the data in the DRS\_MSG\_GETCHGREPLY\_V6 structure.

1. typedef struct {
2. UUID uuidDsaObjSrc;
3. UUID uuidInvocIdSrc;
4. [unique] DSNAME\* pNC;
5. USN\_VECTOR usnvecFrom;
6. USN\_VECTOR usnvecTo;
7. [unique] UPTODATE\_VECTOR\_V2\_EXT\* pUpToDateVecSrc;
8. SCHEMA\_PREFIX\_TABLE PrefixTableSrc;
9. ULONG ulExtendedRet;
10. ULONG cNumObjects;
11. ULONG cNumBytes;
12. [unique] REPLENTINFLIST\* pObjects;
13. BOOL fMoreData;
14. ULONG cNumNcSizeObjects;
15. ULONG cNumNcSizeValues;
16. [range(0,1048576)] DWORD cNumValues;
17. [size\_is(cNumValues)] REPLVALINF\_V3\* rgValues;
18. DWORD dwDRSError;
19. } DRS\_MSG\_GETCHGREPLY\_V9;

**uuidDsaObjSrc:**  [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the server [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**uuidInvocIdSrc:**  [**Invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the server DC.

**pNC:**  The [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) to replicate or the [**FSMO role object**](#gt_6ea17c3e-787a-40e3-a62f-0313dcdc16b7) for an [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6).

**usnvecFrom:**  Data used to correlate calls to IDL\_DRSGetNCChanges.

**usnvecTo:**  Data used to correlate calls to IDL\_DRSGetNCChanges.

**pUpToDateVecSrc:**  [**Stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) filter that describes [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the server has already applied.

**PrefixTableSrc:**  Table for translating [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values in the response to [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

**ulExtendedRet:**  0 or an extended operation error code (section [4.1.10.2.21](#Section_40f60821c7d34d0785563398b1335a7f)).

**cNumObjects:**  Count of items in the **pObjects** linked list.

**cNumBytes:**  Size in bytes of items in or referenced by elements in the **pObjects** linked list.

**pObjects:**  Linked list of [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) updates that the client applies to its [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

**fMoreData:**  False if and only if this is the last response in a [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16).

**cNumNcSizeObjects:**  Estimated number of objects in the server's NC replica.

**cNumNcSizeValues:**  Estimated number of [**link values**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) in the server's NC replica.

**cNumValues:**  Count of items in the **rgValues** array.

**rgValues:**  Link value updates for the client to apply to its NC replica.

**dwDRSError:**  0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

##### DRS\_MSG\_GETCHGREPLY\_NATIVE

The **DRS\_MSG\_GETCHGREPLY\_NATIVE** structure is an alias for the [**DRS\_MSG\_GETCHGREPLY\_V9**](#Section_b9564a194500444ba99b0da1b08cdb6f) data structure.

##### DRS\_MSG\_GETCHGREPLY\_NATIVE\_VERSION\_NUMBER

DRS\_MSG\_GETCHGREPLY\_NATIVE\_VERSION\_NUMBER is a constant. Its value is 9, and it indicates the message version aliased by [**DRS\_MSG\_GETCHGREPLY\_NATIVE**](#Section_8079e22efbc04675979cb95cee7f29a5).

##### COMPRESSED\_DATA

The COMPRESSED\_DATA structure defines a sequence of compressed (if **cbDecompressedSize** ≠ **cbCompressedSize**) or uncompressed (if **cbDecompressedSize** = **cbCompressedSize**) bytes.

1. typedef struct {
2. ULONG cbDecompressedSize;
3. ULONG cbCompressedSize;
4. BYTE data[];
5. } COMPRESSED\_DATA;

**cbDecompressedSize:**  Decompressed size of data.

**cbCompressedSize:**  Compressed size of data.

**data:**  Data stream. The data is padded with zeros, if necessary, so that the block ends on a double word boundary.

##### DRS\_COMP\_ALG\_TYPE

The DRS\_COMP\_ALG\_TYPE enumeration is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for identifying a compression algorithm.

1. typedef enum
2. {
3. DRS\_COMP\_ALG\_NONE = 0,
4. DRS\_COMP\_ALG\_UNUSED = 1,
5. DRS\_COMP\_ALG\_MSZIP = 2,
6. DRS\_COMP\_ALG\_WIN2K3 = 3
7. } DRS\_COMP\_ALG\_TYPE;

**DRS\_COMP\_ALG\_NONE:** No compression.

**DRS\_COMP\_ALG\_UNUSED:** Unused. MUST not be used.

**DRS\_COMP\_ALG\_MSZIP:** [**MSZIP**](#gt_7539ba56-ed97-4cc6-a610-3a1506405c47) algorithm.

**DRS\_COMP\_ALG\_WIN2K3:** Windows Server 2003 compression.

##### DRS\_COMPRESSED\_BLOB

The DRS\_COMPRESSED\_BLOB structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) that results from marshaling a data structure into a byte stream by using [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) and compressing that byte stream.

1. typedef struct {
2. DWORD cbUncompressedSize;
3. DWORD cbCompressedSize;
4. [size\_is(cbCompressedSize)] BYTE\* pbCompressedData;
5. } DRS\_COMPRESSED\_BLOB;

**cbUncompressedSize:**  Size in bytes of the uncompressed byte stream.

**cbCompressedSize:**  Size in bytes of the **pbCompressedData** array.[<29>](#Appendix_A_29" \o "Product behavior note 29)

**pbCompressedData:**  Compressed byte stream.

**Padding**: Data is padded with zeros, if necessary, so that the block ends on an alignment boundary of [LONG](#Section_0fdb03d734b44921b9c46e8025f9e795).

##### ENCRYPTED\_PAYLOAD

The ENCRYPTED\_PAYLOAD packet is the [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a value of an encrypted [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).

1. typedef struct {
2. UCHAR Salt[16];
3. ULONG CheckSum;
4. UCHAR EncryptedData[];
5. } ENCRYPTED\_PAYLOAD;

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| Salt (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CheckSum | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EncryptedData (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Salt (16 bytes):** A 128-bit randomly generated value.

**CheckSum (4 bytes):** A 32-bit [**CRC32**](#gt_9cb45a36-92bb-4c14-b2fd-2ad7e2979bfd) [**checksum**](#gt_fa444149-ef93-4512-a278-2e756295630c) of the data that is encrypted along with the data.

**EncryptedData (variable):** A variable-length byte array that represents the encrypted value.

##### EXOP\_ERR Codes

The following values are error codes for an [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6) request to the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method.

**EXOP\_ERR\_SUCCESS** (0x00000001)

**EXOP\_ERR\_UNKNOWN\_OP** (0x00000002)

**EXOP\_ERR\_FSMO\_NOT\_OWNER** (0x00000003)

**EXOP\_ERR\_UPDATE\_ERR** (0x00000004)

**EXOP\_ERR\_EXCEPTION** (0x00000005)

**EXOP\_ERR\_UNKNOWN\_CALLER** (0x00000006)

**EXOP\_ERR\_RID\_ALLOC** (0x00000007)

**EXOP\_ERR\_FSMO\_OWNER\_DELETED** (0x00000008)

**EXOP\_ERR\_FSMO\_PENDING\_OP** (0x00000009)

**EXOP\_ERR\_MISMATCH** (0x0000000A)

**EXOP\_ERR\_COULDNT\_CONTACT** (0x0000000B)

**EXOP\_ERR\_FSMO\_REFUSING\_ROLES** (0x0000000C)

**EXOP\_ERR\_DIR\_ERROR** (0x0000000D)

**EXOP\_ERR\_FSMO\_MISSING\_SETTINGS** (0x0000000E)

**EXOP\_ERR\_ACCESS\_DENIED** (0x0000000F)

**EXOP\_ERR\_PARAM\_ERR** (0x00000010)

##### EXOP\_REQ Codes

The following values are request codes for [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6).

**EXOP\_FSMO\_REQ\_ROLE** (0x00000001)

**EXOP\_FSMO\_REQ\_RID\_ALLOC** (0x00000002)

**EXOP\_FSMO\_RID\_REQ\_ROLE** (0x00000003)

**EXOP\_FSMO\_REQ\_PDC** (0x00000004)

**EXOP\_FSMO\_ABANDON\_ROLE** (0x00000005)

**EXOP\_REPL\_OBJ** (0x00000006)

**EXOP\_REPL\_SECRETS** (0x00000007)

##### PROPERTY\_META\_DATA

The PROPERTY\_META\_DATA structure contains [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) and [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) information. For more details, see section [4.1.10.5.9](#Section_304ea284592d430d8ab6e9c82a78fa33).

The binary portion of the DNBinary value of the msDS-RevealedUsers attribute contains this structure.

1. typedef struct PROPERTY\_META\_DATA {
2. ATTRTYP attrType;
3. PROPERTY\_META\_DATA\_EXT propMetadataExt;
4. LONGLONG llUnused;
5. } PROPERTY\_META\_DATA;

**attrType:**  The attribute whose value was revealed.

**propMetadataExt:**  The stamp of the revealed attribute value. See PROPERTY\_META\_DATA\_EXT in section [5.155](#Section_aef7ebdec305422495fd585c86b19c38).

**llUnused:** An implementation-specific value. The specific value has no significance.

#### Method-Specific Abstract Types and Procedures

##### AbstractLinkValStampFromConcreteLinkValStamp

1. procedure AbstractLinkValStampFromConcreteLinkValStamp(
2. concreteStamp: VALUE\_META\_DATA\_EXT\_NATIVE) : LinkValueStamp

*Informative summary of behavior*: The AbstractLinkValStampFromConcreteLinkValStamp procedure converts a [**VALUE\_META\_DATA\_EXT\_NATIVE**](#Section_ddd9c5bb779848d582296a51d8dce200) [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) to a [LinkValueStamp](#Section_6a9517897afa47dda96c83fc0e30aa3d).

1. linkValueStamp : LinkValueStamp
2. linkValueStamp := concreteStamp.MetaData
3. linkValueStamp.timeCreated := concreteStamp.timeCreated
4. linkValueStamp.timeExpired := concreteStamp.timeExpired
5. return linkValueStamp

##### AbstractPASFromConcretePAS

1. procedure AbstractPASFromConcretePAS(
2. concretePAS: PARTIAL\_ATTR\_VECTOR\_V1\_EXT,
3. prefixTable: PrefixTable): sequence of ATTRTYP

*Informative summary of behavior*: The AbstractPASFromConcretePAS procedure translates a concrete [PARTIAL\_ATTR\_VECTOR\_V1\_EXT](#Section_1d5c1b34daa44761a8b5d3c0146a0e30) to a sequence of [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983), using *prefixTable* to translate the *concretePAS* entries.

1. abstractPAS: sequence of ATTRTYP
2. i: DWORD
3. for i := 0 to (concretePAS.cAttrs - 1)
4. abstractPAS[i] := LocalAttidFromRemoteAttid(
5. prefixTable, concretePAS.rgPartialAttr[i])
6. endfor
7. return abstractPAS

##### AbstractUTDFromConcreteUTD

1. procedure AbstractUTDFromConcreteUTD(
2. concreteUTD: UPTODATE\_VECTOR\_V2\_EXT): sequence of ReplUpToDateVector

*Informative summary of behavior*: The AbstractUTDFromConcreteUTD procedure translates the [UPTODATE\_VECTOR\_V2\_EXT](#Section_cebd1ccb891b4268b0564b714cdf981e) structure to the [ReplUpToDateVector](#Section_8cb40d62a51d47e39b4e0837edffd61c) [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b).

1. abstractUTD: ReplUpToDateVector
2. for i := 0 to (concreteUTD.length - 1)
3. abstractUTD[i].uuidDsa := concreteUTD.rgCursors[i].uuidDsa
4. abstractUTD[i].usnHighPropUpdate :=
5. concreteUTD.rgCursors[i].usnHighPropUpdate
6. abstractUTD[i].timeLastSyncSuccess :=
7. concreteUTD.rgCursors[i].timeLastSyncSuccess
8. endfor
10. return concreteUTD

##### AttributeAndStamp

1. type AttributeAndStamp = [attribute: ATTRTYP, stamp: AttributeStamp]

This [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) encapsulates the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) of an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) (based on [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).*prefixTable*) and its associated [AttributeStamp](#Section_2973bb80c8ed450da98159639e09820b) on an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

##### AttributeStampCompare

1. procedure AttributeStampCompare(
2. stamp1: AttributeStamp,
3. stamp2: AttributeStamp): integer

*Informative summary of behavior*: The AttributeStampCompare procedure compares two [AttributeStamp](#Section_2973bb80c8ed450da98159639e09820b) values, *stamp1* and *stamp2*. If *stamp1* is greater than *stamp2*, the procedure returns an integer with a value greater than 0. If *stamp1* is equal to *stamp2*, the procedure returns 0. If *stamp1* is less than *stamp2*, then the procedure returns an integer value less than 0. Refer to section 5.11 for details on the comparison of AttributeStamps.

##### ConcretePASFromAbstractPAS

1. procedure ConcretePASFromAbstractPAS(
2. abstractPAS: sequence of ATTRTYP) : PARTIAL\_ATTR\_VECTOR\_V1\_EXT

*Informative summary of behavior*: The ConcretePASFromAbstractPAS procedure translates a sequence of [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) to [PARTIAL\_ATTR\_VECTOR\_V1\_EXT](#Section_1d5c1b34daa44761a8b5d3c0146a0e30). This translation does not require a [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe).

1. concretePAS : PARTIAL\_ATTR\_VECTOR\_V1\_EXT
2. i: DWORD
3. concretePAS.dwVersion := 1
4. concretePAS.dwReserved1 := 0
5. concretePAS.cAttrs := abstractPAS.length
6. for i := 0 to (abstractPAS.length - 1)
7. concretePAS.rgPartialAttr[i] := abstractPAS[i]
8. endfor
9. return concretePAS

##### ConcreteUTDFromAbstractUTD

1. procedure ConcreteUTDFromAbstractUTD(
2. abstractUTD: sequence of ReplUpToDateVector):
3. UPTODATE\_VECTOR\_V1\_EXT

*Informative summary of behavior*: The ConcreteUTDFromAbstractUTD procedure translates a sequence of abstract [ReplUpToDateVector](#Section_8cb40d62a51d47e39b4e0837edffd61c) tuples to [UPTODATE\_VECTOR\_V1\_EXT](#Section_462b424ab50a4c4aa81f48d0f4cf40fe).

1. concreteUTD: UPTODATE\_VECTOR\_V1\_EXT
2. concreteUTD.dwVersion := 1
3. concreteUTD.dwReserved1 := 0
4. concreteUTD.dwReserved2 := 0
5. concreteUTD.cNumCursors := abstractUTD.length
6. for i := 0 to (abstractUTD.length - 1)
7. concreteUTD.rgCursors[i].uuidDsa := abstractUTD[i].uuidDsa
8. concreteUTD.rgCursors[i].usnHighPropUpdate :=
9. abstractUTD[i].usnHighPropUpdate
10. endfor
11. return concreteUTD

##### GetNCChangesNativeReply

1. procedure GetNCChangesNativeReply(
2. replyMessage: DRS\_MSG\_GETCHGREPLY,
3. version: DWORD): DRS\_MSG\_GETCHGREPLY\_NATIVE

*Informative summary of behavior*: The GetNCChangesNativeReply procedure transforms a [DRS\_MSG\_GETCHGREPLY](#Section_65a5cb42c25f4378b06ef87341b21f93) of *version* 1, 6, or 9 to a [DRS\_MSG\_GETCHGREPLY\_NATIVE](#Section_8079e22efbc04675979cb95cee7f29a5) structure. DRS\_MSG\_GETCHGREPLY\_NATIVE is a superset of the data in [**DRS\_MSG\_GETCHGREPLY\_V1**](#Section_bd70a9c3c1d348cf9c24503a5567d09c), [**DRS\_MSG\_GETCHGREPLY\_V6**](#Section_1317a6545dd645ffaf73919cbc7fbb45), and [**DRS\_MSG\_GETCHGREPLY\_V9**](#Section_b9564a194500444ba99b0da1b08cdb6f).

1. msgReplyNative: DRS\_MSG\_GETCHGREPLY\_NATIVE
2. i: DWORD
3. if (version = 1) then
4. msgReplyNative := 0
5. msgReplyNative := replyMessage.V1
7. msgReplyNative.pUpToDateVecSrc^.dwVersion := 2
8. msgReplyNative.pUpToDateVecSrc^.cNumCursors :=
9. replyMessage.V1.pUpToDateVecSrcV1^.cNumCursors
10. for i := 0 to (replyMessage.V1.pUpToDateVecSrcV1^.cNumCursors - 1)
11. msgReplyNative.pUpToDateVecSrc^.rgCursors[i] :=
12. replyMessage.V1.pUpToDateVecSrcV1^.rgCursors[i]
13. msgReplyNative.pUpToDateVecSrc^.rgCursors[i].timeLastSyncSuccess := 0
14. endfor
15. else if (version = 6) then
16. msgReplyNative := 0
17. msgReplyNative := replyMessage.V6
18. msgReplyNative.rgValues :=
19. ReplValInfNativeListFromReplValInfV1List(replyMessage.V6.rgValues)
20. else
21. msgReplyNative = replyMessage.V9
22. endif
23. return msgReplyNative

##### GetStampsForUpdate

1. procedure GetStampsForUpdate(
2. replEntinfList: REPLENTINFLIST,
3. prefixTable: PrefixTable): set of AttributeAndStamp

*Informative summary of behavior*: The GetStampsForUpdate procedure retrieves the [AttributeStamp](#Section_2973bb80c8ed450da98159639e09820b) associated with an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) in the [REPLENTINFLIST](#Section_c38b0412cf004b0cb4f44662a4484a00) [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) and constructs a set of [AttributeAndStamp](#Section_1cf26e6ee5bf45238c839ade3077a14e) tuples.

1. tupleEntry: AttributeAndStamp
2. attrStamps: set of AttributeStamp
3. i: DWORD
4. for i := 0 to (replEntinfList.pMetaDataExt.cNumProps - 1)
5. tupleEntry.attribute := LocalAttidFromRemoteAttid(
6. prefixTable, replEntinfList.Entinf.AttrBlock.pAttr[i].attrTyp)
7. tupleEntry.stamp := AbstractAttrStampFromConcereteAttrStamp(
8. replEntinfList.pMetaDataExt.rgMetaData[i])
9. attrStamps := attrStamps + {tupleEntry}
10. endfor
11. return attrStamps

##### GetWellKnownObject

1. procedure GetWellKnownObject(
2. nc: DSName,
3. guid: GUID): DSName

*Informative summary of behavior*: The GetWellKnownObject procedure returns the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the well-known [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) with the given *guid* in a specified [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

1. attrVals: set of attribute value
2. attrVal: DNBinary
3. attrVals := {nc!wellKnownObjects}
4. for each attrVal in attrVals do
5. if (attrVal.binary = guid) then
6. return attrVal.dn
7. endif
8. endfor
9. return null

##### IsSecretAttribute

1. procedure IsSecretAttribute(attribute : ATTRTYP): boolean

The IsSecretAttribute procedure returns true if *attribute* is an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that contains [**secret data**](#gt_0c8d49b7-bdf7-4824-a91f-481cb10c5052). Otherwise, the procedure returns false.

1. return (attribute in
2. {currentValue, dBCSPwd, initialAuthIncoming, initialAuthOutgoing,
3. lmPwdHistory, ntPwdHistory, priorValue, supplementalCredentials,
4. trustAuthIncoming, trustAuthOutgoing, unicodePwd} )

##### IsUserIncluded

1. procedure IsUserIncluded(
2. userSid: SID
3. groupOrAccountSid: SID)

The IsUserIncluded procedure returns true if *userSid* = *groupOrAccountSid*, or if the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) identified by *userSid* is a member of the set returned by [IDL\_DRSGetMemberships (section 4.1.8)](#Section_d5ace4527cdd4d50bb6439b4c55180a2) with the GroupMembersTransitive option applied to the object identified by *groupOrAccountSid*.

##### ObjAtts

1. type ObjAtts = [obj: DSName, atts: sequence of ATTRTYP]

The ObjAtts [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) encapsulates the identity of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) (*obj*) and a sequence of [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values (*atts*, based on [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).*prefixTable*) for [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of that object.

##### ObjAttVal

1. type ObjAttVal = [obj: DSName; att: ATTRTYP, val: attribute value]

The ObjAttVal [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) encapsulates the identity of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) (*obj*), the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) of an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of that object (*att*, based on [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).*prefixTable*), and a value of that attribute (*val*).

##### PerformModifyDNOperation

1. procedure PerformModifyDNOperation(
2. currentDN: DN,
3. newParentObject: DSName,
4. newRDN: RDN)

The PerformModifyDNOperation procedure performs a Modify DN operation on an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) with the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) **currentDN** by setting its new parent to **newParentObject** and by setting its new [**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) value to **newRDN**. See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.5.4 for more details.

##### RemoveAttrVal

1. procedure RemoveAttrVal(
2. obj: DSName,
3. attr: ATTRTYP,
4. attributeValue: attribute value)

The RemoveAttrVal procedure removes the value *attributeValue* from the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) *attr* on the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) with [DSName](#Section_a0d5477a522946b9890a54b924d487d1) *obj*.

##### SetAttrStamp

1. procedure SetAttrStamp(
2. obj: DSName,
3. attr: ATTRTYP,
4. stamp: AttributeStamp)

The SetAttrStamp procedure sets the [AttributeStamp](#Section_2973bb80c8ed450da98159639e09820b) for the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) *attr* on the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *obj* to *stamp*.

##### SetAttrVal

1. procedure SetAttrVal(
2. obj: DSName,
3. attr: ATTRTYP,
4. attributeValue: attribute value)

The SetAttrVal procedure sets the value *attributeValue* for the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) *attr* on the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *obj*.

##### SetLinkStamp

1. procedure SetLinkStamp(
2. obj: DSName,
3. attr: ATTRTYP,
4. val: attribute value,
5. stamp: AttributeStamp)

The SetLinkStamp procedure sets the [LinkValueStamp](#Section_6a9517897afa47dda96c83fc0e30aa3d) for the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) value *val* on the attribute *attr* on the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *obj* to *stamp*.

#### Client Behavior When Sending the IDL\_DRSGetNCChanges Request

*Informative summary of behavior*: The following three tasks can be accomplished by sending an [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) request to a server:

1. Replicate [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) from the server's [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). The [ReplicateNCRequestMsg](#Section_6f83be5323ed4f07b836b8d0e0a93925) procedure specifies the process of building [DRS\_MSG\_GETCHGREQ](#Section_96affbe17d93453eac759f41c0c94b3b) to perform this task.
2. Replicate a single object from the server's NC replica. The [ReplSingleObjRequestMsg](#Section_ce7af78da7674aeeb5f44e432ae4ee36) procedure specifies the process of building DRS\_MSG\_GETCHGREQ to perform this task.
3. Perform [**extended operations**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6). The [PerformExtendedOpRequestMsg](#Section_96b322bc0e5c4bb7ab744236eea5969d) procedure specifies the process of building DRS\_MSG\_GETCHGREQ to perform this task.

After the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) constructs the request message, it sends the message by using the specified transport: SMTP (as specified in [[MS-SRPL]](%5bMS-SRPL%5d.pdf#Section_ec69eea50d5e428ab5bc66732aaeb866)) if *rf* ≠ null and if *rf.uuidTransport* is the objectGUID of the interSiteTransport object *t*, where *t*!cn = "SMTP"; otherwise, the IP transport ([**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) over TCP).

##### ReplicateNCRequestMsg

1. procedure ReplicateNCRequestMsg(
2. hDrs: DRS\_HANDLE,
3. version: DWORD,
4. nc: DSName,
5. rf: RepsFrom,
6. ulFlags: ULONG,
7. ulMoreFlags: ULONG,
8. cMaxObjects: ULONG,
9. cMaxBytes: ULONG,
10. var msgRequest: DRS\_MSG\_GETCHGREQ)

*Informative summary of behavior*: The client sends an [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) request to a server to replicate the server's changes in an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). The ReplicateNCRequestMsg procedure specifies how the client constructs the request message for this operation.

The procedure has the following arguments:

* *hDrs*: The [DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) that is derived by sending an [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) message to the server.
* *version*: The version number of the input message negotiated between the client and server.
* *nc*: The [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the root of the NC replica that is to be replicated.
* *rf*: The [RepsFrom](#Section_3ef27d3cb9c944048e53ebf3a64a9a10) that corresponds to the server from which to replicate.
* *ulFlags*: Zero or more of the following bit flags. The client MUST supply the same flags for each request in a given [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16), with the exception of DRS\_ADD\_REF, DRS\_GET\_ANC, DRS\_USE\_COMPRESSION, and DRS\_GET\_NC\_SIZE.
  + **DRS\_ADD\_REF**: Requests that the server add an entry to the repsTo [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) for the client on the root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of the NC replica that is being replicated.
  + **DRS\_WRIT\_REP**: Indicates that the client contains (or is constructing) a full, [**writable NC replica**](#gt_51db485c-dcf6-4845-99b3-2df414ef0aa9).
  + **DRS\_ASYNC\_REP**: Requests that the server send only the root object of the NC replica.
  + **DRS\_CRITICAL\_ONLY**: Signals the server not to send objects *o* where *o*!isCriticalSystemObject is absent or *o*!isCriticalSystemObject is false.
  + **DRS\_GET\_ANC**: Signals the server to send all [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) for each [**ancestor object**](#gt_874fa4dc-37f4-4467-91c3-78d5e4e5a410) of object *o* before sending updates for object *o*.
  + **DRS\_GET\_NC\_SIZE**: Signals the server to set *cNumNcSizeObjects* in *pmsgOut* to an estimate of the number of objects in its NC replica.
  + **DRS\_FULL\_SYNC\_PACKET**: Requests that the server send all attributes of the objects in its reply, rather than sending only the updated attributes.
  + **DRS\_SYNC\_FORCED**: Signals the server to honor the request even if its [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) has otherwise been disabled.
  + **DRS\_USE\_COMPRESSION**: Requests that the server reply by using one of the compressed reply versions ([DRS\_MSG\_GETCHGREPLY\_V2](#Section_677d8fab6aa143279b6f62a6ad7fcfa3) or [DRS\_MSG\_GETCHGREPLY\_V7](#Section_5ef4f597a3974f6fa98b7a034247d886)).
  + **DRS\_SYNC\_PAS**: Indicates replication of additional attributes to the partial [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) already present on the client.
  + **DRS\_SPECIAL\_SECRET\_PROCESSING**: Requests that the server not ship attribute values of attributes that contain [**secret data**](#gt_0c8d49b7-bdf7-4824-a91f-481cb10c5052). Servers prior to Windows Server 2008 operating system ignore this flag.
  + **DRS\_GET\_ALL\_GROUP\_MEMBERSHIP**: Requests that the server ship all [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) membership. If this flag is not specified, the server ships only [**universal group**](#gt_f46053d6-0708-4094-ac63-57c1bcb73d32) membership. Servers prior to Windows Server 2008 ignore this flag.
  + **DRS\_REF\_GCSPN**: Requests that the server add an entry to repsTo for the client on the root object of the NC replica that is being replicated. When repsTo is set using this flag, the notifying client [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) contacts the server DC using the [**service principal name**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) that begins with "GC" (section [2.2.3.2](#Section_41efc56e00074e88bafed7af61efd91f)).
* *ulMoreFlags*: Zero or more of the following bit flags. The client MUST supply the same flags for each request in a given replication cycle, with the exception of DRS\_GET\_TGT.
  + **DRS\_GET\_TGT**: Signals the server to send all updates for the [**target object**](#gt_62c95f88-0024-410c-b008-b637d04803ad) of a [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) update before sending the link value update.
* *cMaxObjects*: Recommended limit on the number of objects to include in the reply.
* *cMaxBytes*: Recommended limit on the number of bytes to include in the reply.
* *msgRequest*: The procedure populates corresponding fields in this structure depending on the value that is passed in the *version* parameter.

1. msgIn: DRS\_MSG\_GETCHGREQ\_V10
2. msgRequest: DRS\_MSG\_GETCHGREQ
3. prefixEntry: PrefixTableEntry
4. partialAttrSetSeq: sequence of DSName
5. schemaSignature: sequence of BYTE
6. ncType: ULONG
7. /\* NTDSDSA\_OPT\_DISABLE\_INBOUND\_REPL defined in
8. \* [MS-ADTS] section 6.1.1.2.2.1.2.1.1, "nTDSDSA Object"\*/
9. if NTDSDSA\_OPT\_DISABLE\_INBOUND\_REPL in DSAObj()!options and
10. not DRS\_SYNC\_FORCED in ulFlags then
11. return ERROR\_DS\_DRA\_SINK\_DISABLED
12. endif
13. if IsAdlds() and ServerExtensions(hDrs).ConfigObjGUID ≠ NULLGUID and ServerExtensions(hDrs).ConfigObjGUID ≠ ConfigNC().GUID then
14. return ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS;
15. endif
16. msgIn.ulMoreFlags := ulMoreFlags
17. msgIn.cMaxObjects := cMaxObjects
18. msgIn.cMaxBytes := cMaxBytes
19. msgIn.ulExtendedOp := 0
20. msgIn.uuidDsaObjDest := dc.serverGuid
21. msgIn.pNC := ADR(nc)
22. msgIn.liFsmoInfo := 0
23. if (ObjExists(nc)) then
24. msgIn.pUpToDateVecDest :=
25. ConcreteUTDFromAbstractUTD(nc!replUpToDateVector)
26. else
27. msgIn.pUpToDateVecDest := null
28. endif
29. /\* Fill usnvecFrom and uuidInvocIdSrc fields.
30. \* usnvecFrom: This field contains the value of the usnVec field in
31. \* RepsFrom tuple corresponding to the IDL\_DRSGetNCChanges server
32. \* DC, or zeros if no such repsFrom is present.
33. \* uuidInvocIdSrc: If the usnvecFrom field is not zeros, this field
34. \* MUST contain the uuidInvocId from the same tuple from which the
35. \* usnVec field was retrieved. Otherwise, this field contains
36. \* zeros.\*/
37. if (rf = null) then
38. msgIn.usnvecFrom := 0
39. msgIn.uuidInvocIdSrc := 0
40. else
41. msgIn.usnvecFrom := rf.usnVec
42. msgIn.uuidInvocIdSrc := rf.uuidInvocId
43. endif
44. if AmIRODC()then
45. if DRS\_WRIT\_REP in ulFlags then
46. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
47. endif
48. ext := ServerExtensions(hDrs)
49. if not DRS\_EXT\_LH\_BETA2 in ext.dwFlags and
50. msgIn.pNC^ = SchemaNC() then
51. ulFlags := ulFlags + {DRS\_WRIT\_REP}
52. endif
53. endif
54. ncType = GetNCType(nc)
55. if not NCT\_GC\_PARTIAL in ncType then
56. ulFlags := ulFlags + {DRS\_GET\_ALL\_GROUP\_MEMBERSHIP}
57. endif
58. msgIn.ulFlags := ulFlags
59. if (DRS\_WRIT\_REP in ulFlags) or
60. (not DRS\_SYNC\_PAS in ulFlags) then
61. msgIn.pPartialAttrSetEx := null
62. else
63. msgIn.pPartialAttrSetEx := ConcretePASFromAbstractPAS(rf.pasData)
64. endif
65. /\* set msgIn.pPartialAttrSet field \*/
66. if ObjExists(nc) and nc!partialAttributeSet ≠ null then
67. msgIn.pPartialAttrSet := ConcretePASFromAbstractPAS(
68. nc!partialAttributeSet)
69. else
70. if (NCT\_GC\_PARTIAL in ncType and
71. NCT\_FILTERED\_ATTRIBUTE\_SET in ncType)} then
72. msgIn.pPartialAttrSet := FilteredGCPAS()
73. else if NCT\_FILTERED\_ATTRIBUTE\_SET in ncType then
74. msgIn.pPartialAttrSet := FilteredPAS()
75. else if NCT\_GC\_PARTIAL in ncType then
76. msgIn.pPartialAttrSet := GCPAS()
77. else
78. msgIn.pPartialAttrSet := null
79. endif
80. endif
81. msgIn.PrefixTableDest = ConcretePTFromAbstractPT(dc.prefixTable)
82. /\* Add schema signature to msgIn.PrefixTableDest \*/
83. schemaSignature := SchemaInfo()
84. prefixEntry.ndx := 0
85. prefixEntry.prefix.length := schemaSignature.length
86. prefixEntry.prefix.element := elements of schemaSignature
87. Append prefixEntry to msgIn.PrefixTableDest.pPrefixEntry
88. msgIn.PrefixTableDest.PrefixCount :=
89. msgIn.PrefixTableDest.PrefixCount + 1
90. if version = 5 then
91. msgRequest.V5 := msgIn
92. msgRequest.V5.pUpToDateVecDestV1 := msgIn.pUpToDateVecDest
93. else if version = 8 then
94. msgRequest.V8 := msgIn
95. else
96. msgRequest.V10 := msgIn
97. endif

##### ReplSingleObjRequestMsg

1. procedure ReplSingleObjRequestMsg(
2. hDrs: DRS\_HANDLE,
3. version: DWORD,
4. nc: DSName,
5. object: DSName,
6. rf: RepsFrom,
7. ulFlags: ULONG,
8. ulMoreFlags: ULONG,
9. cMaxObjects: ULONG,
10. cMaxBytes: ULONG,
11. fWithSecrets: boolean,
12. var msgRequest: DRS\_MSG\_GETCHGREQ): DWORD

*Informative summary of behavior*: The client can send an [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) request to the server to replicate changes from a single [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). The ReplSingleObjRequestMsg procedure specifies how the request message is constructed for this operation. The arguments for this method are the same as those for the procedure [ReplicateNCRequestMsg](#Section_6f83be5323ed4f07b836b8d0e0a93925), with the following exceptions:

* *object*: The [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the object that should be replicated.
* *fWithSecrets*: The object's secret [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) should be replicated. Only [**RODCs**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870) need to make, and can make, this request.

The procedure returns a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if it cannot construct *msgRequest*.

1. msgRequest: DRS\_MSG\_GETCHGREQ
2. msgIn: DRS\_MSG\_GETCHGREQ\_V10
3. ncType: ULONG
4. /\* An NC replica with root of DSName nc must already exist on the
5. client \*/
6. if (not PartialGCReplicaExists(nc) and
7. not FullReplicaExists(nc)) then
8. return ERROR\_DS\_DRA\_BAD\_NC
9. endif
10. /\* Only RODCs are allowed to request secrets explicitly \*/
11. if fWithSecrets and not AmIRODC() then
12. return ERROR\_INVALID\_PARAMETER
13. endif
14. if fWithSecrets then
15. msgIn.ulExtendedOp := EXOP\_REPL\_SECRETS
16. else
17. msgIn.ulExtendedOp := EXOP\_REPL\_OBJ
18. endif
19. if AmIRODC()then
20. if DRS\_WRIT\_REP in ulFlags then
21. return ERROR\_INVALID\_PARAMETER
22. endif
23. ext := ServerExtensions(hDrs)
24. if not DRS\_EXT\_LH\_BETA2 in ext.dwFlags and
25. msgIn.pNC^ = SchemaNC() then
26. ulFlags := ulFlags + {DRS\_WRIT\_REP}
27. endif
28. endif
29. ncType = GetNCType(nc)
30. if not NCT\_GC\_PARTIAL in ncType then
31. ulFlags := ulFlags + {DRS\_GET\_ALL\_GROUP\_MEMBERSHIP}
32. endif
33. msgIn.ulFlags := ulFlags
34. msgIn.ulMoreFlags := ulMoreFlags
35. msgIn.cMaxObjects := cMaxObjects
36. msgIn.cMaxBytes := cMaxBytes
37. msgIn.uuidDsaObjDest := dc.serverGuid
38. msgIn.pNC := ADR(object)
39. msgIn.liFsmoInfo := 0
40. msgIn.pUpToDateVecDest :=
41. ConcreteUTDFromAbstractUTD(nc!replUpToDateVector)
42. msgIn.pPartialAttrSetEx := null
43. /\* set msgIn.pPartialAttrSet field \*/
44. if ObjExists(nc) and nc!partialAttributeSet ≠ null then
45. msgIn.pPartialAttrSet := ConcretePASFromAbstractPAS(
46. nc!partialAttributeSet)
47. else
48. if (NCT\_GC\_PARTIAL in ncType and
49. NCT\_FILTERED\_ATTRIBUTE\_SET in ncType)} then
50. msgIn.pPartialAttrSet := FilteredGCPAS()
51. else if NCT\_FILTERED\_ATTRIBUTE\_SET in ncType then
52. msgIn.pPartialAttrSet := FilteredPAS()
53. else if NCT\_GC\_PARTIAL in ncType then
54. msgIn.pPartialAttrSet := GCPAS()
55. else
56. msgIn.pPartialAttrSet := null
57. endif
58. endif
59. msgIn.PrefixTableDest = ConcretePTFromAbstractPT(dc.prefixTable)
60. /\* Fill usnvecFrom and uuidInvocIdSrc fields.
61. \* usnvecFrom: This field contains the value of the usnVec field in
62. \* RepsFrom tuple corresponding to the IDL\_DRSGetNCChanges server
63. \* DC, or zeros if no such repsFrom is present.
64. \* uuidInvocIdSrc: If the usnvecFrom field is not zeros, this field
65. \* MUST contain the uuidInvocId from the same tuple from which the
66. \* usnVec fieldwas retrieved. Otherwise, this field contains
67. \* zeros.\*/
68. if (rf = null) then
69. msgIn.usnvecFrom := 0
70. msgIn.uuidInvocIdSrc := 0
71. else
72. msgIn.usnvecFrom := rf.usnVec
73. msgIn.uuidInvocIdSrc := rf.uuidInvocId
74. endif
75. if version = 5 then
76. msgRequest.V5 := msgIn
77. msgRequest.V5.pUpToDateVecDestV1 := msgIn.pUpToDateVecDest
78. else if version = 8 then
79. msgRequest.V8 := msgIn
80. else
81. msgRequest.V10 := msgIn
82. endif
83. return 0

##### PerformExtendedOpRequestMsg

1. procedure PerformExtendedOpRequestMsg (
2. hDrs: DRS\_HANDLE,
3. version: DWORD,
4. nc: DSName,
5. roleOwnerObject: DSName,
6. rf: RepsFrom,
7. ulFlags: ULONG,
8. ulMoreFlags: ULONG,
9. ulExtendedOp: ULONG,
10. cMaxObjects: ULONG,
11. cMaxBytes: ULONG,
12. var msgRequest: DRS\_MSG\_GETCHGREQ): DWORD

*Informative summary of behavior*: A client sends an [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) request to a server to perform an [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6). The procedure PerformExtendedOpRequestMsg specifies how the request message is constructed for this operation.

The arguments for this method are the same as those for the procedure [ReplicateNCRequestMsg](#Section_6f83be5323ed4f07b836b8d0e0a93925), with the following exceptions:

* *ulExtendedOp*: The requested extended operation. The client MUST supply the same value of this field for each request in a given [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16). The possible values are:
  + EXOP\_FSMO\_REQ\_ROLE, for a [**FSMO role owner**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b) transfer.
  + EXOP\_FSMO\_REQ\_RID\_ALLOC, for a [**RID**](#gt_df3d0b61-56cd-4dac-9402-982f1fedc41c) allocation from the RID Master FSMO role owner.
  + EXOP\_FSMO\_RID\_REQ\_ROLE, for transfer of the RID Master [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f).
  + EXOP\_FSMO\_REQ\_PDC, for transfer of the [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) FSMO role.
  + EXOP\_FSMO\_ABANDON\_ROLE, to request the server to request an extended operation role transfer from the client.
* *roleOwnerObject*: The client sets this value based on the value of *ulExtendedOp*, as per the following table:

| ulExtendedOp | roleOwnerObject |
| --- | --- |
| EXOP\_FSMO\_REQ\_ROLE | The [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the [**FSMO role object**](#gt_6ea17c3e-787a-40e3-a62f-0313dcdc16b7). |
| EXOP\_FSMO\_REQ\_RID\_ALLOC | The value of the rIDManagerReference [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of [DefaultNC](#Section_d262eef5f2594628968041f1b799ad23)(). |
| EXOP\_FSMO\_RID\_REQ\_ROLE | The value of the rIDManagerReference attribute of DefaultNC(). |
| EXOP\_FSMO\_REQ\_PDC | DefaultNC(). |
| EXOP\_FSMO\_ABANDON\_ROLE | The DSName of the FSMO role object. |

The procedure returns a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if it not able to construct *msgRequest*.

1. msgIn: DRS\_MSG\_GETCHGREQ\_V10
2. serverObj: DSName
3. computerObj: DSName
4. ridSetReferences: DSName
5. /\* An NC replica with root nc must already exist on the client \*/
6. if (not MasterReplicaExists(nc)) then
7. return ERROR\_DS\_DRA\_BAD\_NC
8. endif
9. msgIn.ulFlags := ulFlags
10. msgIn.ulMoreFlags := ulMoreFlags
11. msgIn.cMaxObjects := cMaxObjects
12. msgIn.cMaxBytes := cMaxBytes
13. msgIn.ulExtendedOp := ulExtendedOp
14. msgIn.uuidDsaObjDest := dc.serverGuid
15. msgIn.pNC := ADR(roleOwnerObject)
16. msgIn.pUpToDateVecDest :=
17. ConcreteFromAbstractUTD(nc!replUpToDateVector)
18. msgIn.pPartialAttrSetEx := null
19. msgIn.pPartialAttrSet := null
20. msgIn.PrefixTableDest := 0
21. if (ulExtendedOp = EXOP\_FSMO\_REQ\_RID\_ALLOC) then
22. serverObj := DSAObj()!parent
23. computerObj := serverObject!serverReference
24. ridSetReferences := computerObj!ridSetReferences
25. if ((not ridSetReferences = null) and
26. (ridSetReferences!isDeleted = false)) and
27. (not ridSetReferences!rIDNextRid = null) and
28. (not ridSetReferences!rIDNextRid = 0) and
29. (not ridSetReferences!rIDAllocationPool = null)) then
30. msgIn.liFsmoInfo := ridSetReferences!rIDAllocationPool
31. else
32. msgIn.liFsmoInfo := 0
33. endif
34. else
35. msgIn.liFsmoInfo := 0
36. endif
37. /\* Fill usnvecFrom and uuidInvocIdSrc fields.
38. \* usnvecFrom: This field contains the value of the usnVec field in
39. \* RepsFrom tuple corresponding to the IDL\_DRSGetNCChanges server
40. \* DC, or zeros if no such repsFrom is present.
41. \* uuidInvocIdSrc: If the usnvecFrom field is not zeros, this field
42. \* MUST contain the uuidInvocId from the same tuple from which the
43. \* usnVec field was retrieved. Otherwise, this field contains
44. \* zeros.\*/
45. if (rf = null) then
46. msgIn.usnvecFrom := 0
47. msgIn.uuidInvocIdSrc := 0
48. else
49. msgIn.usnvecFrom := rf.usnVec
50. msgIn.uuidInvocIdSrc := rf.uuidInvocId
51. endif
52. if version = 5 then
53. msgRequest.V5 := msgIn
54. msgRequest.V5.pUpToDateVecDestV1 := msgIn.pUpToDateVecDest
55. else if version = 8 then
56. msgRequest.V8 := msgIn
57. else
58. msgRequest.V10 := msgIn
59. endif
60. return 0

#### Server Behavior of the IDL\_DRSGetNCChanges Method

*Informative summary of behavior*: The IDL\_DRSGetNCChanges method returns a response for a single request in a [**cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16).

This method is invoked through the drsuapi [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface. It is also invoked as a local procedure for requests that are received using the SMTP transport ([[MS-SRPL]](%5bMS-SRPL%5d.pdf#Section_ec69eea50d5e428ab5bc66732aaeb866)).

1. ULONG
2. IDL\_DRSGetNCChanges(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_GETCHGREQ \*pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DRS\_MSG\_GETCHGREPLY \*pmsgOut)
10. err: ULONG
11. msgIn: DRS\_MSG\_GETCHGREQ\_V10
12. ncRoot: DSName
13. obj: DSName
14. msgOut: DRS\_MSG\_GETCHGREPLY\_NATIVE
15. schemaSignature: sequence of BYTE
16. prefixEntry: PrefixTableEntry
17. responseSmtpAddress: unicodestring
18. fullReplicaFlags: set of integer
19. fullReplicaRequest : boolean
20. ValidateDRSInput(hDrs, 3)
21. pdwOutVersion^ := 1
22. pmsgOut^ := 0
23. err := TransformInput(hDrs, dwInVersion, pmsgIn^, msgIn,
24. pdwOutVersion, responseSmtpAddress)
25. if err ≠ 0 then
26. return err
27. endif
28. /\* Perform access checks. \*/
29. if msgIn.pNC = null then
30. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
31. endif
32. ncRoot := GetObjectNC(msgIn.pNC^)
33. if ncRoot = null then
34. return ERROR\_DS\_CANT\_FIND\_EXPECTED\_NC
35. endif
36. if IsGetNCChangesPermissionGranted(msgIn) == FALSE then
37. return ERROR\_DRA\_ACCESS\_DENIED
38. endif
39. /\* Validate inputs. \*/
40. obj := msgIn.pNC^
41. if AmILHServer() = false then
42. /\* Downlevel OS does not understand
43. DRS\_SPECIAL\_SECRET\_PROCESSING flags. They just ignore it.
44. \*/
45. msgIn.ulFlags := msgIn.ulFlags - {DRS\_SPECIAL\_SECRET\_PROCESSING}
46. endif
47. if msgIn.ulExtendedOp = 0 then
48. /\* Validate normal replication request. \*/
49. if not FullReplicaExists(obj) and not PartialGCReplicaExists(obj)
50. then
51. return ERROR\_DS\_CANT\_FIND\_EXPECTED\_NC
52. endif
53. else
54. /\* Validate extended operation request. \*/
55. if not ObjExists(obj) then
56. return ERROR\_DS\_CANT\_FIND\_EXPECTED\_NC
57. endif
58. endif
59. if AmILHServer() then
60. if (msgIn.pPartialAttrSet = null and
61. msgIn.pPartialAttrSetEx = null) then
62. fullReplicaRequest := true
63. else
64. fullReplicaRequest := false
65. endif
66. else
67. if (DRS\_WRITE\_REP in msgIn.ulFlags) then
68. fullReplicaRequest := true
69. else
70. fullReplicaRequest := false
71. endif
72. endif
73. if (fullReplicaRequest) then
74. /\* Validate Full Replica request. \*/
75. if not IT\_WRITE in obj!instanceType then
76. return ERROR\_DRA\_SOURCE\_IS\_PARTIAL
77. endif
78. if DRS\_SYNC\_PAS in msgIn.ulFlags then
79. return ERROR\_INVALID\_PARAMETER
80. endif
81. else
82. /\* Validate Partial Replica request. \*/
83. if msgIn.pPartialAttrSet = null
84. or msgIn.pPartialAttrSet.cAttrs = 0 then
85. return ERROR\_INVALID\_PARAMETER
86. endif
87. if DRS\_SYNC\_PAS in msgIn.ulFlags and
88. (msgIn.pPartialAttrSetEx = null
89. or msgIn.pPartialAttrSetEx.cAttrs = 0) then
90. return ERROR\_INVALID\_PARAMETER
91. endif
92. if msgIn.PrefixTableDest.PrefixCount = 0 then
93. return ERROR\_INVALID\_PARAMETER
94. endif
95. endif
96. if IT\_NC\_GOING in ncRoot!instanceType
97. /\* NC replica is no longer accepting requests. \*/
98. return ERROR\_DRA\_NO\_REPLICA
99. endif
100. if msgIn.uuidInvocIdSrc ≠ DSAObj()!invocationId then
101. msgIn.usnvecFrom := 0
102. endif
103. /\* Construct response. \*/
104. if msgIn.ulExtendedOp = 0 then
105. /\* Perform normal replication. \*/
106. err := GetReplChanges(hDrs, null, null, msgIn, msgOut)
107. else
108. /\* Perform extended operation. Errors are returned in
109. \* msgOut.ulExtendedErr. \*/
110. ProcessFsmoRoleRequest(hDrs, msgIn, msgOut)
111. err := 0
112. endif
113. if err = 0 then
114. msgOut.pNC := msgIn.pNC
115. msgOut.usnvecFrom := msgIn.usnvecFrom
116. msgOut.uuidDsaObjSrc := dc.serverGuid
117. msgOut.PrefixTableSrc := ConcretePTFromAbstractPT(dc.prefixTable)
118. msgOut.uuidInvocIdSrc := DSAObj()!invocationId
119. /\* Sort msgOut.rgValues into ascending order. \*/
120. SortResponseLinks(msgOut)
122. /\* Add schema signature to msgOut.PrefixTableSrc. \*/
123. schemaSignature := SchemaInfo()
124. prefixEntry.ndx := 0
125. prefixEntry.prefix.length := schemaSignature.length
126. prefixEntry.prefix.element := elements of schemaSignature
127. Append prefixEntry to msgOut.PrefixTableSrc.pPrefixEntry
128. msgOut.PrefixTableSrc.PrefixCount :=
129. msgOut.PrefixTableSrc.PrefixCount+1
130. err := TransformOutput(msgOut, msgIn.ulFlags, pdwOutVersion^,
131. pmsgOut)
132. endif
133. if responseSmtpAddress ≠ null then
134. Send the response using using the SMTP transport to
135. responseSmtpAddress
136. endif
137. return err

##### TransformInput

1. procedure TransformInput(
2. hDrs: DRS\_HANDLE,
3. requestVersion: DWORD,
4. requestUnion: DRS\_MSG\_GETCHGREQ,
5. var nativeRequest: DRS\_MSG\_GETCHGREQ\_V10,
6. pdwOutVersion: ADDRESS OF DWORD,
7. var responseSmtpAddress: unicodestring): ULONG

*Informative summary of behavior*: The TransformInput procedure transforms the received request message into a V10 request, which is a superset of the supported request messages.

1. optionalFeatures: sequence of msDS-OptionalFeature objects /\* [MS-ADSC] \*/
2. extClient: DRS\_EXTENSIONS\_INT
3. extServer: DRS\_EXTENSIONS\_INT
4. optionalFeatureBit: integer
5. extClient := ClientExtensions(hDrs)
6. responseSmtpAddress := null
7. if requestVersion < dc.minimumGetChangesRequestVersion
8. return ERROR\_REVISION\_MISMATCH
9. if requestVersion = 10
10. /\* Windows Server 2008 R2 RPC request. \*/
11. nativeRequest := requestUnion.V10
12. if DRS\_EXT\_GETCHGREPLY\_V9 in extClient.dwFlags then
13. pdwOutVersion^ = 9
14. else if DRS\_EXT\_GETCHGREPLY\_V6 in extClient.dwFlags then
15. pdwOutVersion^ = 6
16. else
17. return ERROR\_REVISION\_MISMATCH
18. endif
19. else if requestVersion = 8
20. /\* Windows Server 2003 RPC request. \*/
21. nativeRequest := requestUnion.V8
22. nativeRequest.ulMoreFlags := 0
23. if not DRS\_EXT\_GETCHGREPLY\_V6 in extClient.dwFlags then
24. return ERROR\_REVISION\_MISMATCH
25. else
26. pdwOutVersion^ := 6
27. endif
28. else if requestVersion = 7 then
29. /\* Windows Server 2003 SMTP request. \*/
30. responseSmtpAddress := requestUnion.V7.pmtxReturnAddress^.mtx\_name
31. nativeRequest := requestUnion.V7.V3
32. nativeRequest.pUpToDateVecDest :=
33. requestUnion.V7.V3.pUpToDateVecDestV1
34. nativeRequest.pPartialAttrSet := requestUnion.V7.pPartialAttrSet
35. nativeRequest.pPartialAttrSetEx :=
36. requestUnion.V7.pPartialAttrSetEx
37. nativeRequest.PrefixTableDest := requestUnion.V7.PrefixTableDest
38. nativeRequest.ulMoreFlags := 0
39. if not DRS\_EXT\_GETCHGREPLY\_V6 in extClient.dwFlags then
40. return ERROR\_REVISION\_MISMATCH
41. else
42. pdwOutVersion^ := 6
43. endif
44. else if requestVersion = 5 then
45. /\* Windows 2000 RPC request. \*/
46. nativeRequest := requestUnion.V5
47. nativeRequest.pUpToDateVecDest :=
48. requestUnion.V5.pUpToDateVecDestV1
49. nativeRequest.pPartialAttrSetEx := null
50. nativeRequest.PrefixTableDest :=
51. ConcretePTFromAbstractPT(dc.prefixTable)
52. nativeRequest.ulMoreFlags := 0
53. if ({DRS\_WRIT\_REP} ∩ requestUnion.V5.ulFlags) = null then
54. nativeRequest.pPartialAttrSet := GCPAS()
55. endif
56. pdwOutVersion^ := 1
57. else if requestVersion = 4 then
58. /\* Windows 2000 SMTP request. \*/
59. responseSmtpAddress := requestUnion.V4.pmtxReturnAddress^.mtx\_name
60. nativeRequest := requestUnion.V4.V3
61. nativeRequest.pUpToDateVecDest :=
62. requestUnion.V4.V3.pUpToDateVecDestV1
63. if ({DRS\_WRIT\_REP} ∩ requestUnion.V4.V3.ulFlags) = null then
64. nativeRequest.pPartialAttrSet := GCPAS()
65. endif
66. nativeRequest.pPartialAttrSetEx := null
67. nativeRequest.ulMoreFlags := 0
68. pdwOutVersion^ := 1
69. else
70. /\* Unsupported request. \*/
71. return ERROR\_REVISION\_MISMATCH
72. endif
73. if ({DRS\_WRIT\_REP} ∩ nativeRequest.ulFlags) ≠ null then
74. nativeRequest.ulFlags :=
75. nativeRequest.ulFlags + {DRS\_GET\_ALL\_GROUP\_MEMBERSHIP}
76. endif
77. if (responseSmtpAddress = null) ≠ (not DRS\_MAIL\_REP in
78. nativeRequest.ulFlags) then
79. return ERROR\_INVALID\_PARAMETER
80. endif
81. extServer := ServerExtensions(hDrs)
82. optionalFeatures := list of msDS-OptionalFeature objects from the
83. Optional Features container /\* [MS-ADTS] section 6.1.1.2.4.1.3 \*/
84. foreach feature in optionalFeatures
85. if (GetOptionalFeatureBit(feature!msDS-OptionalFeatureGUID, optionalFeatureBit))
86. if (optionalFeatureBit in extServer.dwFlagsEx) and
87. optionalFeatureBit not in extClient.dwExtCaps then
88. /\* Feature is enabled on the server but the client
89. is not capable of supporting it \*/
90. return ERROR\_REVISION\_MISMATCH
91. endif
92. endif
93. /\* NOTE: The behavior of the server is undefined if \*/
94. /\* procedure GetOptionalFeatureBit() returns false. \*/
95. endfor
96. return 0

##### GetReplChanges

1. procedure GetReplChanges(
2. hDrs: DRS\_HANDLE,
3. searchFilter: LDAPString,
4. dirSyncFlags: ULONG,
5. msgIn: DRS\_MSG\_GETCHGREQ\_V10,
6. var msgOut: DRS\_MSG\_GETCHGREPLY\_NATIVE): ULONG

*Informative summary of behavior*: The GetReplChanges procedure processes an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) Search request with LDAP\_SERVER\_DIRSYNC\_OID control or a normal [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) request; that is, an [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) request that is not a [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f) request. It adds changed [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) and [**link values**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) to the response, subject to the scope (*msgIn.pNC^*, *msgIn.ulFlags*), filter criteria (*msgIn.pUpToDateVecDest*, *msgIn.ulFlags*, *msgIn.pPartialAttrSet*, *msgIn.pPartialAttrSetEx*, *searchFilter*, *dirSyncFlags*), response limits (*msgIn.cMaxObjects*, *msgIn.cMaxBytes*), and the previous server cookie (*msgIn.usnvecFrom*) in the request. It returns 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

1. err: ULONG
2. ncRoot: DSName
3. pUtd: ADDRESS OF UPTODATE\_VECTOR\_V1\_EXT
4. scope: set of DSName
5. attribute: ATTRTYP
6. partialAttrs: set of ATTRTYP
7. partialAttrsEx: set of ATTRTYP
8. changedObjs: set of ObjAtts
9. changedLinks: set of ObjAttVal
10. responseObjs: set of ObjAtts
11. responseLinks: set of ObjAttVal
12. anc: ObjAtts
13. clientDSA : DSName
14. updRefs: DRS\_MSG\_UPDREFS\_V1 /\* See IDL\_DRSUpdateRefs structures. \*/
15. tgt: DSName
16. if AmIRODC() then
17. return ERROR\_DS\_DRA\_SOURCE\_DISABLED
18. endif
19. /\* check whether outbound replication is disabled \*/
20. /\* NTDSDSA\_OPT\_DISABLE\_OUTBOUND\_REPL defined in
21. \* [MS-ADTS] section 6.1.1.2.2.1.2.1.1, "nTDSDSA Object" \*/
22. if NTDSDSA\_OPT\_DISABLE\_OUTBOUND\_REPL in DSAObj()!options and
23. not DRS\_SYNC\_FORCED in msgIn.ulFlags and
24. not dirSyncFlags then
25. return ERROR\_DS\_DRA\_SOURCE\_DISABLED
26. endif
27. ncRoot := GetObjectNC(msgIn.pNC^)
28. /\* Determine stamp filter to apply to the response. \*/
29. if DRS\_FULL\_SYNC\_PACKET in msgIn.ulFlags then
30. pUtd := null
31. else
32. pUtd := msgIn.pUpToDateVecDest
33. endif
34. /\* Determine attribute filters to apply to the response. \*/
35. if msgIn.pPartialAttrSet = null
36. partialAttrs := null
37. else
38. partialAttrs := {}
39. foreach id in msgIn.pPartialAttrSet
40. attribute := LocalAttidFromRemoteAttid(msgIn.PrefixTableDest, id)
41. if (not IT\_WRITE in ncRoot!instanceType) and
42. (not attribute in ncRoot!partialAttributeSet) then
43. return ERROR\_DS\_DRA\_INCOMPATIBLE\_PARTIAL\_SET
44. endif
45. partialAttrs := partialAttrs + { attribute }
46. endfor
47. endif
48. if msgIn.pPartialAttrSetEx = null
49. partialAttrsEx := null
50. else
51. partialAttrsEx := {}
52. foreach id in msgIn.pPartialAttrSetEx
53. attribute := LocalAttidFromRemoteAttid(msgIn.PrefixTableDest, id)
54. if (not IT\_WRITE in ncRoot!instanceType) and
55. (not attribute in ncRoot!partialAttributeSet) then
56. return ERROR\_DS\_DRA\_INCOMPATIBLE\_PARTIAL\_SET
57. endif
58. partialAttrsEx:= partialAttrsEx + { attribute }
59. endfor
60. endif
61. /\* Get nTDSDSA of the client \*/
62. clientDSA := select one o from ConfigNC() where
63. o!objectGUID = msgIn.uuidDsaObjDest
64. /\* Get the set of all objects that are in scope. \*/
65. scope := GetReplScope(msgIn, searchFilter)
66. /\* Get object and link value changes in scope. \*/
67. GetChangesInScope(scope, pUtd, msgIn.ulExtendedOp, partialAttrs,
68. partialAttrsEx, dirSyncFlags, changedObjs, changedLinks)
69. /\* Choose subsets of changedObjs and changedLinks to include in this
70. \* response. Set usnvecTo and fMoreData in out to indicate the
71. \* subset to return in the next response, if any. \*/
72. GetResponseSubset(msgIn, changedObjs, changedLinks, msgOut,
73. responseObjs, responseLinks)
74. /\* Add responseObjs to response. \*/
75. foreach o in responseObjs
76. if DRS\_GET\_ANC in msgIn.ulFlags then
77. /\* Ancestors predicate: insert any changes to parent before any
78. \* changes to child. \*/
79. foreach n in Ancestors of o.obj, most distant ancestor first
80. anc := select one a from changedObjs where a.obj = n
81. if anc ≠ null then
82. err := AddObjToResponse(
83. hDrs, anc, ncRoot, msgIn.ulFlags, 0, clientDSA, msgOut)
84. if err ≠ 0 then
85. return err
86. endif
87. endif
88. endfor
89. endif
90. err := AddObjToResponse(
91. hDrs, o, ncRoot, msgIn.ulFlags, 0, clientDSA, msgOut)
92. if err ≠ 0 then
93. return err
94. endif
95. endfor
96. /\* Add responseLinks to response. \*/
97. foreach v in responseLinks
98. if DRS\_GET\_ANC in msgIn.ulFlags then
99. /\* Ancestors predicate: insert any changes to object before any
100. \* changes to its link values. \*/
101. anc := select one a from changedObjs where a.obj = v.obj
102. if anc ≠ null then
103. err := AddObjToResponse(hDrs, anc, ncRoot,
104. msgIn.ulFlags, 0, clientDSA, msgOut)
105. if err ≠ 0 then
106. return err
107. endif
108. endif
109. endif
110. if DRS\_GET\_TGT in msgIn.ulMoreFlags then
111. /\* Target predicate: insert any changes to the target object
112. \* before any changes to the link value. \*/
113. tgt := GetDSNameFromAttrVal(v.att, v.val)
114. if DRS\_GET\_ANC in msgIn.ulFlags then
115. /\* Ancestors predicate: insert any changes to the ancestors of
116. \* the target before any changes to the target. \*/
117. foreach n in Ancestors of tgt, most distant ancestor first
118. anc := select one a from changedObjs where a.obj = n
119. if anc ≠ null then
120. err := AddObjToResponse(hDrs, anc, ncRoot,
121. msgIn.ulFlags, 0, clientDSA, msgOut)
122. if err ≠ 0 then
123. return err
124. endif
125. endif
126. endfor
127. endif
128. err := AddObjToResponse(hDrs, tgt, ncRoot,
129. msgIn.ulFlags, 0, clientDSA, msgOut)
130. if err ≠ 0 then
131. return err
132. endif
133. endif
134. AddLinkToResponse(v, msgIn, msgOut)
135. endfor
136. if not msgOut.fMoreData
137. msgOut.pUpToDateVecSrc := The cycle goal, as specified in
138. section 4.1.10.1.2.
139. endif
140. if DRS\_GET\_NC\_SIZE in msgIn.ulFlags then
141. msgOut.cNumNcSizeObjects := Approximate number of objects in
142. NC replica msgIn.pNC^
143. msgOut.cNumNcSizeValues := Approximate number of link values
144. with stamps in NC replica msgIn.pNC^
145. endif
146. if (DRS\_ADD\_REF in msgIn.ulFlags and msgIn.uuidDsaObjDest ≠ NULLGUID) then
147. /\* Client has requested the server to add a repsTo entry. \*/
148. updRefs.uuidDsaDes := msgIn.uuidDsaObjDest
149. updRefs.pNC := msgIn.pNC^
150. updRefs.pszDsaDest := NetworkAddress of DC corresponding to
151. msgIn.uuidDsaObjDest
152. updRefs.ulOptions := {DRS\_ADD\_REF, DRS\_ASYNC\_OP,
153. DRS\_GETCHG\_CHECK} +
154. { msgIn.ulFlags ∩ {DRS\_WRIT\_REP, DRS\_REF\_GCSPN}}
155. /\* Using updRefs, perform repsTo add to the specified NC replica,
156. \* the result value is a Windows error code or 0.
157. err := UpdateRefs(updRefs^.V1)
158. if(err ≠ 0) then
159. return err
160. endif
161. endif
162. return 0

##### GetReplScope

1. procedure GetReplScope(
2. msgIn: DRS\_MSG\_GETCHGREQ\_V10,
3. searchFilter: LDAPString): set of DSName

*Informative summary of behavior*: The GetReplScope procedure returns the set of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) considered for normal [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) or for an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) Search request with LDAP\_SERVER\_DIRSYNC\_OID control: the objects in the requested [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) (*msgIn.pNC^*) or a subset thereof, as indicated by the request flags (*msgIn.ulFlags*) and the search filter (*searchFilter*). If the DRS\_ASYNC\_REP request flag is specified, the subset includes only the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root. If the DRS\_CRITICAL\_ONLY request flag is specified, the subset includes only those objects with isCriticalSystemObject = true and their ancestors.

1. scope: set of DSName
2. ncRoot: DSName
3. anc: DSName
4. ncRoot := GetObjectNC(msgIn.pNC^)
5. if DRS\_ASYNC\_REP in msgIn.ulFlags then
6. if (ObjectMatchesSearchFilter(ncRoot, searchFilter) = true) then
7. scope := {ncRoot}
8. endif
9. else if DRS\_CRITICAL\_ONLY in msgIn.ulFlags then
10. scope := select all o from subtree-ts-included ncRoot where
11. o!isCriticalSystemObject = true
12. foreach o in scope
13. foreach anc in Ancestors of o
14. if not anc in scope then
15. if (ObjectMatchesSearchFilter(anc, searchFilter) = true) then
16. scope := scope + {anc}
17. endif
18. endif
19. endif
20. endfor
21. else
22. scope := select all o from subtree-ts-included ncRoot where true
23. foreach o in ncRoot!subRefs
24. if (ObjectMatchesSearchFilter(o, searchFilter) = true) then
25. scope := scope + {o}
26. endif
27. endfor
28. endif
29. return scope

##### ObjectMatchesSearchFilter

1. procedure ObjectMatchesSearchFilter (
2. o: DSNAME,
3. searchFilter: LDAPString) : boolean

This procedure returns true if the search filter (*searchFilter*) is null. If the search filter is not null, it returns true if the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) whose DSNAME is "*o*" matches the search filter; otherwise it returns false. See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) for search filter processing, specifically section 3.1.1.3.1.3.1, Search Filters.

##### GetChangesInScope

1. procedure GetChangesInScope(
2. scope: set of DSName,
3. pUtd: ADDRESS OF UPTODATE\_VECTOR\_V1\_EXT,
4. ulExtendedOp: DWORD,
5. partialAttrs: set of ATTRTYP,
6. partialAttrsEx: set of ATTRTYP,
7. dirSyncFlags: ULONG,
8. var changedObjs: set of ObjAtts,
9. var changedLinks: set of ObjAttVal)

*Informative summary of behavior*: The GetChangesInScope procedure inspects the [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in *scope* and returns the object and [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) that are sent to the client over the course of the [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16) or as a result of processing [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) Search request with LDAP\_SERVER\_DIRSYNC\_OID control, as determined by the [**up-to-date vector**](#gt_42564a26-2ae7-41a2-a67c-3c74381d8538) (*pUtd*), the [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6) (*ulExtendedOp*), flags (*dirSyncFlags)* associated with the LDAP\_SERVER\_DIRSYNC\_OID control, and the partial [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) filters (*partialAttrs* and *partialAttrsEx*).

1. o: DSName
2. a: ATTRTYP
3. attrsFound: set of ATTRTYP
4. attrsReq: set of ATTRTYP
5. stamp: AttributeStamp
6. cursor: UPTODATE\_CURSOR\_V2
7. /\* Get the set of objects in scope with attribute stamps that the
8. \* client did not have knowledge of at the beginning of this
9. \* cycle. \*/
10. changedObjs := {}
11. foreach o in scope
12. attrsFound := {}
13. attrsReq := {}
14. foreach a of o's object class
15. stamp := AttrStamp(o, a)
16. if stamp ≠ null and
17. ((ulExtendedOp = EXOP\_REPL\_SECRETS and IsSecretAttribute(a))
18. or not FilterAttribute(o, a, stamp, pUtd,
19. partialAttrs, partialAttrsEx, dirSyncFlags)) then
20. attrsFound := attrsFound + {a}
21. endif
22. if(a = instanceType or a = proxiedObjectName) then
23. attrsReq := attrsReq + {a}
24. endif
25. endfor
26. if attrsFound ≠ {} then
27. changedObjs := changedObjs + [obj: o, atts: attrsFound + attrsReq]
28. else if (IT\_NC\_HEAD in o!instanceType and pUtd ≠ null)
29. stamp := AttrStamp(o, uSNChanged)
30. cursor := select one c from pUtd^.rgCursors where c.uuidDsa =
31. stamp.uuidOriginating
32. if cursor = null or cursor.usnHighPropUpdate < stamp.usnOriginating
33. then
34. changedObjs := changedObjs + [obj: o, atts: attrsFound + attrsReq]
35. endif
36. endif
37. endfor
38. /\* Get the set of link values in scope with stamps that the client
39. \* did not have knowledge of at the beginning of this cycle. \*/
40. if (GetForestFunctionalLevel() ≥ 1 or dc.fLinkValueStampEnabled = true) then
41. changedLinks := {}
42. foreach o in scope
43. foreach a in Link Attributes of o's object class
44. foreach v in GetAttVals(o, a, true)
45. stamp := LinkStamp(o, a, v)
46. /\* If v was last updated in win2k forest mode
47. \* then it does not have LinkValueStamp associated with it.
48. \* LinkStamp() returns null in that case and this value will
49. \* not be added to changedLinks.
50. \*/
51. if stamp ≠ null
52. and not FilterAttribute(o, a, stamp, pUtd,
53. partialAttrs, partialAttrsEx, dirSyncFlags) then
54. changedLinks := changedLinks + [obj: o, att: a, val: v]
55. endif
56. endfor
57. endfor
58. endfor
59. endif

##### FilterAttribute

1. procedure FilterAttribute(
2. o: DSName,
3. attribute: ATTRTYP,
4. s: AttributeStamp,
5. pUtd: ADDRESS OF UPTODATE\_VECTOR\_V1\_EXT,
6. partialAttrs: set of ATTRTYP,
7. partialAttrsEx: set of ATTRTYP,
8. dirSyncFlags: ULONG): boolean

*Informative summary of behavior*: The FilterAttribute procedure determines whether an [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) ([**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) or [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238)) that is in scope should be filtered out of the set of changes to send in the [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16). The rules are as follows:

* If the client's [**up-to-date vector**](#gt_42564a26-2ae7-41a2-a67c-3c74381d8538) *pUtd* asserts that the client has already applied the update with [**stamps**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00), the update is filtered out, provided that *attribute* is not in the *partialAttrsEx* set. The elements of *partialAttrsEx* are not subject to filtering by the up-to-date vector.
* If *partialAttrs* is not null (indicating the client has a partial replica) and *attribute* is not in *partialAttrs* + *partialAttrsEx*, then the update is filtered out.
* If *partialAttrs* is not null, *attribute* is member, *o* is of [**class**](#gt_18393bbe-0c06-42b7-890d-b94a9a40b6e0) group, and *o* is not a [**universal group**](#gt_f46053d6-0708-4094-ac63-57c1bcb73d32), then the update is filtered out.
* If *attribute* is the naming attribute (that is, cn for [**objects of class**](#gt_c2c67596-8d8f-42b9-9c70-1c4f7c952200) container, as shown below) for the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) class of *o*, the update is filtered out.
* If LDAP\_DIRSYNC\_OBJECT\_SECURITY is in *dirSyncFlags*, and the client does not have access rights to read the object, all the updates are filtered out except updates to the isDeleted and isRecycled attributes.

1. filtered: boolean
2. cursor: UPTODATE\_CURSOR\_V2
3. filtered := false
4. if pUtd ≠ null and partialAttrsEx ≠ null
5. and not attribute in partialAttrsEx then
6. /\* Filter updates with stamps that the client's up-to-date vector
7. \* asserts the client has already applied to its NC replica.
8. \*/
9. cursor := select one c from pUtd^.rgCursors where c.uuidDsa =
10. s.uuidOriginating
11. if cursor ≠ null and cursor.usnHighPropUpdate >= s.usnOriginating
12. then
13. filtered := true
14. endif
15. endif
16. if not filtered and partialAttrs ≠ null then
17. /\* Filter updates to attributes that are not in the client's
18. \* partial replica.
19. \*/
20. if not attribute in partialAttrs + partialAttrsEx then
21. filtered := true
22. endif
23. endif
24. if not filtered and partialAttrs ≠ null and attribute = member then
25. /\* Filter updates to the member attribute from the client's
26. \* partial replica if the group is not a universal group.
27. \*/
28. if group in o!objectClass and
29. not GROUP\_TYPE\_UNIVERSAL\_GROUP in o!groupType then
30. filtered := true
31. endif
32. endif
33. if not filtered then
34. /\* Filter updates to the naming attribute of o. \*/
35. if attribute = o!rdnType then
36. filtered := true
37. endif
38. endif
39. if not filtered then
40. /\* Filter non replicated attributes of o. \*/
41. if AttrIsNonReplicated(attribute) then
42. filtered := true
43. endif
44. endif
45. if not filtered then
46. /\* If LDAP\_DIRSYNC\_OBJECT\_SECURITY in dirSyncFlags, and the client does
47. not have access rights to read the object, all the updates are filtered
48. out except updates to isDeleted and isRecycled attributes. \*/
49. if LDAP\_DIRSYNC\_OBJECT\_SECURITY in dirSyncFlags and
50. (AccessCheckObject(o, RIGHT\_DS\_LIST\_OBJECT) = false or
51. AccessCheckObject(o.parent, RIGHT\_DS\_LIST\_CONTENTS) = false) and
52. attribute ≠ isDeleted and
53. attribute ≠ isRecycled then
54. filtered := true
55. endif
56. endif
57. return filtered

##### GetResponseSubset

1. procedure GetResponseSubset(
2. msgIn: DRS\_MSG\_GETCHGREQ\_V10,
3. changedObjs: set of ObjAtts,
4. changedLinks: set of ObjAttVal,
5. var msgOut: DRS\_MSG\_GETCHGREPLY\_NATIVE,
6. var responseObjs: set of ObjAtts,
7. var responseLinks: set of ObjAttVal)

The GetResponseSubset procedure selects subsets of the changed [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) and [**link values**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) to include in this response. It utilizes the cookie msgIn.usnvecFrom—which is zero or a value returned to the client in a previous response—and the client-requested limits *msgIn.cMaxObjects* and *msgIn.cMaxBytes* to determine the subsets. This procedure then sets msgOut.usnvecTo to a new cookie that the client presents in its next request as msgIn.usnvecFrom.

The server SHOULD[<30>](#Appendix_A_30" \o "Product behavior note 30) choose a subset such that the response will contain no more objects than *msgIn.cMaxObjects* and no more bytes (before any compression is applied) than *msgIn.cMaxBytes*.

It is valid for the response to contain no objects or link values. It is valid for an object or a link value to appear multiple times in a single response, and the object's [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) values or the link values need not be identical. It is valid for an object or a link value to appear both in the current response and in an earlier response in the same [**cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16), and the object's attribute values or the link values need not be identical.

If the server determines, by using state that is maintained via msgIn.usnvecFrom and msgOut.usnvecTo, that inclusive of what it is sending in this response, it will have sent at least *changedObjs* and *changedLinks* to the client, then it concludes the cycle by returning with msgOut.fMoreData = false. Therefore, if this is the first response message of a cycle, the server only returns with msgOut.fMoreData = false if *responseObjs* = *changedObjs* and *responseLinks* = *changedLinks*.

Subject to resource constraints on the server, if neither *changedObjs* nor *changedLinks* increases during a sequence of calls, the server eventually returns msgOut.fMoreData = false.

##### AddObjToResponse

1. procedure AddObjToResponse(
2. hDrs: DRS\_HANDLE,
3. o: ObjAtts,
4. ncRoot: DSName,
5. ulFlags: set of integer,
6. ulExtendedOp: DWORD,
7. clientDSA: DSName,
8. var msgOut: DRS\_MSG\_GETCHGREPLY\_NATIVE) : ULONG

*Informative summary of behavior*: The AddObjToResponse procedure constructs a [REPLENTINFLIST](#Section_c38b0412cf004b0cb4f44662a4484a00) structure for a changed [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) and appends it to the response.

1. err: ULONG
2. re: REPLENTINFLIST
3. pAttr: ADDRESS OF ATTR
4. attribute: ATTRTYP
5. attrObj: DSName
6. attrVals: sequence of attribute values
7. i: DWORD
8. j: DWORD
9. err := 0
10. /\* Construct a REPLENTINFLIST to represent the changes. \*/
11. re := all zeros
12. re.fIsNcPrefix := (o.obj = ncRoot)
13. if name in o.atts and not re.fIsNcPrefix then
14. re.pParentGuid := ADR(o.obj!parent)
15. endif
16. re.EntInf.pName := ADR(o.obj)
17. re.EntInf.AttrBlock.pAttrs := array of ATTR of size o.atts.length
18. re.EntInf.AttrBlock.attrCount := o.atts.length
19. re.pMetaDataExt := PROPERTY\_META\_DATA\_EXT\_VECTOR
20. with rgMetaData of size o.atts.length
21. re.pMetaDataExt^.cNumProps := o.atts.length
22. for i := 0 to o.atts.length - 1
23. attribute := o.atts[i]
24. attrObj := SchemaObj(attribute)
25. re.pMetaDataExt^.rgMetaData[i] = AttrStamp(o.obj, attribute)
26. pAttr := ADR(re.EntInf.AttrBlock.pAttrs[i])
27. pAttr^.attrTyp := attribute
28. pAttr^.AttrVal.valCount := 0
29. if AmILHServer() and
30. DRS\_SPECIAL\_SECRET\_PROCESSING in ulFlags and
31. IsSecretAttribute(attribute) then
32. /\* secret attribute, send a null value \*/
33. pAttr^.AttrVal.pAVal = null
34. re.pMetaDataExt^.rgMetaData[i].timeChanged = 0
35. else if not AmILHServer() and /\* W2K3 or lower \*/
36. not DRS\_WRIT\_REP in ulFlags and /\* partial replication \*/
37. IsSecretAttribute(attribute) then
38. /\* secret attribute in W2K3 or lower servers, send a null value \*/
39. pAttr^.AttrVal.pAVal = null
40. re.pMetaDataExt^.rgMetaData[i].timeChanged = 0
41. else
42. /\* not special processing \*/
43. attrVals := GetAttrVals(o, attribute, false)
44. pAttr^.AttrVal.pAVal := ARRAY OF ATTRVAL
45. WITH SIZE attrVals.length
46. for j := 0 to attrVals.length - 1
47. /\* If attribute is a link value attribute, then add it to the
48. \* response here only if it does not have a LinkValueStamp
49. \* associated with it. This can happen if the current forest
50. \* functional level is DS\_BEHAVIOR\_WIN2000 or the attribute
51. \* value attrVals[j] was last updated when the forest
52. \* functional level was DS\_BEHAVIOR\_WIN2000. If the
53. \* attribute value has a LinkValueStamp associated with it,
54. \* then it will be sent in the response packet by method
55. \* AddLinkToResponse. Forest functional levels are listed
56. \* in [MS-ADTS] section 6.1.4.4,
57. \* "msDS-Behavior-Version: Forest Functional Level".
58. \*/
59. if (attrObj!linkID = null) or
60. ((attrObj!linkID ≠ null) and
61. (LinkStamp(o.obj, attribute, attrVals[j]) = null) then
62. pAttr^.AttrVal.pAVal[j] := ATTRVALFromValue(
63. attrVals[j], Syntax(attribute), dc.prefixTable)
64. pAttr^.AttrVal.valCount := pAttr^.AttrVal.valCount + 1
65. endif
66. endfor /\* j := \*/
67. endif
68. err := EncryptValuesIfNecessary(hDrs, pAttr^)
69. if err ≠ 0 then
70. return err
71. endif
72. /\* if secrets are being sent to RODC then log it to revealed
73. \* list \*/
74. if (EXOP\_REPL\_SECRETS in ulExtendedOp) then
75. UpdateRevealedList(clientDSA, o.obj, attribute)
76. endif
77. endfor /\* i := \*/
78. /\* Add re to the response. \*/
79. Add re to the end of the linked list msgOut.pObjects
80. msgOut.cNumObjects := msgOut.cNumObjects + 1
81. return err

##### UpdateRevealedList

1. procedure UpdateRevealedList(
2. rodcDSA: DSName,
3. revealedObject: DSName,
4. attribute: ATTRTYP)

*Informative summary of behavior*: The UpdateRevealedList procedure adds or [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) an entry for the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) *attribute* of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *revealedObject* on the msDS-RevealedUsers attribute of the computer object that corresponds to the nTDSDSA object *rodcDSA*. The msDS-RevealedUsers attribute is of type DNBinary. The binary portion of the *attribute* value contains a [PROPERTY\_META\_DATA](#Section_ab1ad92035384d6491976e700ef1f222) structure in its binary form. The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) portion of *attribute* value contains *revealedObject*.

1. serverObj: DSName
2. computerObj: DSName
3. attrSchemaObj: DSName
4. revealedObjectsNew: set of DNBinary
5. obj: DNBinary
6. propMetadata: PROPERTY\_META\_DATA
7. propMetadataCurrent: PROPERTY\_META\_DATA
8. newRevealedObjectVal: DNBinary
9. /\* Revealed list has entries only for secret attributes \*/
10. if not IsSecretAttribute(attribute) then
11. return
12. endif
13. /\* Get the computer object corresponding to nTDSDSA object rodcDSA \*/
14. serverObj := rodcDSA!parent
15. computerObj := serverObj!serverReference
16. /\* filter superseded entries from the msDS-RevealedUsers set \*/
17. revealedObjectsNew := {}
18. foreach obj in computerObj!msDS-RevealedUsers
19. propMetadata := loophole(obj.binary, PROPERTY\_META\_DATA)
20. if (obj.object\_dn ≠ revealedObject) or
21. (propMetaData.attrType ≠ attribute) or
22. (StampCompare(propMetaData.propMetadataExt,
23. AttrStamp(revealedObject, attribute) > 0) then
24. revealedObjectsNew := revealedObjectsNew + { obj }
25. endif
26. endfor
27. /\* add the new entry to the set \*/
28. propMetadataCurrent.attrType := attribute
29. propMetadataCurrent.propMetadataExt :=
30. AttrStamp(revealedObject, attribute)
31. propMetadataCurrent.llUnused := An implementation-specific value
32. that is of no significance to the protocol.
33. newRevealedObjectVal.binary :=
34. loophole(propMetadataCurrent, sequence of byte)
35. newRevealedObjectVal.object\_dn:= revealedObject
36. revealedObjectsNew := revealedObjectsNew + { newRevealedObjectVal }
37. /\* set attribute value to new set \*/
38. computerObj!msDS-RevealedUsers := revealedObjectsNew

##### AddLinkToResponse

1. procedure AddLinkToResponse(
2. v: ObjAttVal,
3. msgIn: DRS\_MSG\_GETCHGREQ\_V10,
4. var msgOut: DRS\_MSG\_GETCHGREPLY\_NATIVE)

*Informative summary of behavior*: The AddLinkToResponse procedure constructs a [REPLVALINF\_NATIVE](#Section_882a0aa8fb564be6ad4ab9030314111e) structure for a changed [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) and appends it to the response.

1. rv: REPLVALINF\_NATIVE
2. rvs: sequence of REPLVALINF\_NATIVE
3. stamp: LinkValueStamp
4. filterGroups: boolean
5. filterGroups := true;
6. if AmILHServer() then
7. if DRS\_GET\_ALL\_GROUP\_MEMBERSHIP in msgIn.ulFlags then
8. filterGroups := false
9. endif
10. else
11. if DRS\_WRITE\_REP in msgIn.ulFlags then
12. filterGroups := false
13. endif
14. endif
15. if filterGroups = true and
16. group in v.obj!objectClass and
17. not GROUP\_TYPE\_UNIVERSAL\_GROUP in v.obj!groupType and
18. v.att = member
19. /\* non-universal group membership is replicated out unless
20. explicitly requested \*/
21. return
22. endif
23. /\* Construct a REPLVALINF\_NATIVE to represent the changes to send. \*/
24. rv.pObject = v.obj
25. rv.attrType := v.att
26. rv.AVal := ATTRVALFromValue(v.val, Syntax(v.att), dc.prefixTable)
27. stamp := LinkStamp(v.obj, v.att, v.val)
28. rv.fIsPresent := stamp.timeDeleted = 0
29. rv.MetaData := stamp
30. /\* Add rv to the response. \*/
31. if msgOut.cNumValues ≠ 0
32. Copy elements from msgOut.rgValues to rvs
33. endif
34. rvs[msgOut.cNumValues] := rv
35. msgOut.rgValues := elements of rvs
36. msgOut.cNumValues := msgOut.cNumValues + 1

##### EncryptValuesIfNecessary

1. procedure EncryptValuesIfNecessary(
2. hDrs: DRS\_HANDLE,
3. var attr: ATTR) : ULONG

*Informative summary of behavior*: The EncryptValuesIfNecessary procedure encrypts the values of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that contain [**secret data**](#gt_0c8d49b7-bdf7-4824-a91f-481cb10c5052). It performs the encryption by using an MD5 [**digest**](#gt_1c222b9e-7176-4840-9d19-e65895b9fc62) (as specified in [[RFC1321]](https://go.microsoft.com/fwlink/?LinkId=90275)), a [**CRC32**](#gt_9cb45a36-92bb-4c14-b2fd-2ad7e2979bfd) [**checksum**](#gt_fa444149-ef93-4512-a278-2e756295630c) (as specified in [[ISO/IEC 13239]](https://go.microsoft.com/fwlink/?LinkId=98149)), and an RC4 stream cipher (as specified in [[RC4]](https://go.microsoft.com/fwlink/?LinkId=93759)). This encryption is in addition to the encryption that is provided by [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) privacy.

1. sessionKey: sequence of BYTE
2. i: integer
3. salt: sequence of BYTE
4. md5Context: MD5\_CTX
5. crc: ULONG
6. pPayload: ADDRESS OF ENCRYPTED\_PAYLOAD
7. if not IsSecretAttribute(attr.attrTyp) then
8. /\* No additional encryption necessary. \*/
9. return 0
10. endif
11. if not DRS\_EXT\_STRONG\_ENCRYPTION in ClientExtensions(hDrs).dwFlags then
12. return SEC\_E\_ALGORITHM\_MISMATCH
13. endif
14. /\* Get session key associated with the RPC connection. \*/
15. sessionKey := session key associated with security context of hDrs,
16. as specified by [MS-RPCE] section 3.3.1.5.2, "Building and Using a
17. Security Context", and [MS-KILE] section 3.1.1.2, "Cryptographic
18. Material"
19. /\* Encrypt each value of this attribute. \*/
20. for i := 0 to attr.AttrVal.valCount - 1
21. salt := randomly generated 128-bit number
22. /\* Calculate checksum of the clear value. \*/
23. crc := CRC32 [ISO/IEC 13239] of the attr.AttrVal.pAVal[i].valLen
24. bytes starting at attr.AttrVal.pAVal[i].pVal
25. /\* Compute encryption key. \*/
26. MD5Init(md5Context)
27. MD5Update(md5context, sessionKey, sessionKey.length)
28. MD5Update(md5context, salt, 16)
29. MD5Final(md5Context)
30. /\* Construct payload, encrypting its contents with the exception of
31. \* the Salt field. \*/
32. pPayload := New ENCRYPTED\_PAYLOAD, sized to hold
33. attr.AttrVal.pAVal[i].valLen bytes in the EncryptedData field
34. pPayload^.Salt := salt
35. pPayload^.Checksum := crc
36. Copy attr.AttrVal.pAVal[i].valLen bytes from
37. attr.AttrVal.pAVal[i].pVal to pPayload^.EncryptedData
38. Encrypt attr.AttrVal.pAVal[i].valLen + 4 bytes starting at the
39. address of pPayload^.Checksum using the RC4 stream cipher
40. algorithm [RC4] with encryption key md5Context.digest
41. /\* Replace the clear value with the encrypted value. \*/
42. attr.AttrVal.pAVal[i].pVal := pPayload
43. attr.AttrVal.pAVal[i].valLen := attr.AttrVal.pAVal[i].valLen + 20
44. endfor
45. return 0

##### ProcessFsmoRoleRequest

1. procedure ProcessFsmoRoleRequest(
2. hDrs: DRS\_HANDLE,
3. msgIn: DRS\_MSG\_GETCHGREQ\_V10,
4. var msgOut: DRS\_MSG\_GETCHGREPLY\_NATIVE)

*Informative summary of behavior*: The ProcessFsmoRoleRequest procedure performs the requested [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f) operation indicated by *msgIn.ulExtendedOp*.

1. fsmoObj: DSName
2. clientDsaObj: DSName
3. serverObj: DSName: DSName
4. rodcObj: DSNAME: DSName
5. clientComputerObj: DSName
6. clientRidSetObj: DSName
7. ownerDsaObj: DSName
8. scope: set of DSName
9. ridAllocLoHi: ULONGLONG
10. ridAllocHi: DWORD
11. ridReqHi: DWORD
12. ridAvailLoHi: ULONGLONG
13. ridAvailLo: DWORD
14. ridAvailHi: DWORD
15. changedObjs: set of ObjAtts
16. changedLinks: set of ObjAttVal
17. /\* Specific error check when at DC functional level Win2K3 \*/
18. if (DSAObj()!msDS-Behavior-Version = DS\_BEHAVIOR\_WIN2003) and
19. (not DRS\_WRIT\_REP in msgIn.ulFlags) then
20. msgOut.ulExtendedRet := EXOP\_ERR\_PARAM\_ERR
21. return
22. endif
23. fsmoObj := msgIn.pNC^
24. if not ObjExists(fsmoObj) then
25. msgOut.ulExtendedRet := EXOP\_ERR\_UPDATE\_ERR
26. return
27. endif
28. if msgIn.uuidDsaObjDest = null then
29. msgOut.ulExtendedRet := EXOP\_ERR\_UPDATE\_ERR
30. return
31. endif
32. clientDsaObj := select one o from ConfigNC()where
33. o!objectGUID = msgIn.uuidDsaObjDest
34. if clientDsaObj = null then
35. msgOut.ulExtendedRet := EXOP\_ERR\_UNKNOWN\_CALLER
36. return
37. endif
38. scope := {}if msgIn.ulExtendedOp in {EXOP\_FSMO\_REQ\_ROLE, EXOP\_FSMO\_REQ\_PDC,
39. EXOP\_FSMO\_RID\_REQ\_ROLE} then
40. /\* Change the FSMO role owner from the server to the client. \*/
41. if fsmoObj!fSMORoleOwner ≠ DSAObj() then
42. msgOut.ulExtendedRet := EXOP\_ERR\_FSMO\_NOT\_OWNER
43. return
44. endif
45. fsmoObj!fSMORoleOwner := clientDsaObj
46. scope := GetRoleScope(fsmoObj)
47. else if msgIn.ulExtendedOp = EXOP\_FSMO\_ABANDON\_ROLE then
48. /\* Request a change in the FSMO role owner from the current owner
49. \* to the server. The server will refuse to take the FSMO role if
50. \* it is not a full replica and cannot own FSMO. \*/
51. if AmIRODC() then
52. msgOut.ulExtendedRet := EXOP\_ERR\_FSMO\_REFUSING\_ROLES
53. endif
54. if fsmoObj!fSMORoleOwner ≠ DSAObj() then
55. ownerDsaObj := fsmoObj!fSMORoleOwner
56. if not ObjExists(ownerDsaObj) then
57. msgOut.ulExtendedRet := EXOP\_ERR\_UNKNOWN\_CALLER
58. return
59. else if ownerDsaObj!isDeleted = true
60. msgOut.ulExtendedRet := EXOP\_ERR\_FSMO\_OWNER\_DELETED
61. return
62. endif
63. Call IDL\_DRSGetNCChanges as a client to the server identified by
64. ownerDsaObj to perform a EXOP\_FSMO\_REQ\_ROLE extended
65. operation; see the client request generation and response
66. processing sections
67. if fsmoObj!fSMORoleOwner ≠ DSAObj() then
68. /\* Transfer failed. \*/
69. msgOut.ulExtendedRet := EXOP\_ERR\_COULDNT\_CONTACT
70. return
71. endif
72. endif
73. else if msgIn.ulExtendedOp = EXOP\_FSMO\_REQ\_RID\_ALLOC then
74. /\* Allocate a block of RIDs for the client DC. \*/
75. if fsmoObj ≠ DefaultNC()!rIDManagerReference then
76. msgOut.ulExtendedRet := EXOP\_ERR\_MISMATCH
77. return
78. else if fsmoObj!fSMORoleOwner ≠ DSAObj() then
79. msgOut.ulExtendedRet := EXOP\_ERR\_FSMO\_NOT\_OWNER
80. return
81. endif
82. /\* Locate or create the RID Set object for the client DC. \*/
83. serverObj := clientDsaObj!parent
84. clientComputerObj := serverObj!serverReference
85. if clientComputerObj!rIDSetReferences = null then
86. clientRidSetObj := An implementation-defined DSName in the
87. default NC such that not ObjExists(clientRidSetObj)
88. Create object with DSName clientRidSetObj such that
89. rIDSet in clientRidSetObj!objectClass
90. /\* Windows Behavior: Windows sets clientRidSetObj to be a child
91. \* of clientComputerObj. \*/
92. clientComputerObj!rIDSetReferences := clientRidSetObj
93. else
94. clientRidSetObj := clientComputerObj!rIDSetReferences
95. endif
96. /\* Get the current RID allocation for the client DC. \*/
97. ridAllocLoHi := clientRidSetObj!rIDAllocationPool
98. ridAvailHi := most significant 32 bits of ridAvailLoHi
99. ridReqHi := most significant 32 bits of msgIn.liFsmoInfo
100. if ridAllocLoHi = 0 or ridAvailHi = 0 or ridReqHi ≥ ridAvailHi then
101. /\* The client DC has indeed exhausted its current allocation,
102. \* according to our records. \*/
103. /\* Get the range of RIDs that have not yet been allocated to any
104. \* DC. \*/
105. ridAvailLoHi := fsmoObj!rIDAvailablePool
106. ridAvailLo := least significant 32 bits of ridAvailLoHi
107. ridAvailHi := most significant 32 bits of ridAvailLoHi
108. /\* Select a subset of the unallocated RIDs and allocate them to
109. \* the client. \*/
110. Assign a value to ridAllocHi according to any implementation-
111. defined policy such that ridAvailLo < ridAllocHi < ridAvailHi.
112. /\* Windows Behavior: By default, Windows sets ridAllocHi to
113. \* ridAvailLo + 500. \*/
114. ridAllocLoHi := ridAvailLo as least significant 32 bits and
115. ridAllocHi as most significant 32 bits
116. ridAvailLo := ridAllocHi + 1
117. ridAvailLoHi := ridAvailLo as least significant 32 bits and
118. ridAvailHi as most significant 32 bits
119. fsmoObj!rIDAvailablePool := ridAvailLoHi
120. clientRidSetObj!rIDAllocationPool := ridAllocLoHi
121. clientRidSetObj!rIDPreviousAllocationPool := 0
122. clientRidSetObj!rIDNextRID := 0
123. /\* Windows Behavior: rIDUsedPool [MS-ADA3] is not used anywhere,
124. \* but Windows always sets it to zero. \*/
125. clientRidSetObj!rIDUsedPool := 0
126. msgOut.liFsmoInfo := ridAllocLoHi
127. endif
128. scope := GetRoleScope(fsmoObj) +
129. {clientComputerObj, clientRidSetObj}
130. else if EXOP\_REPL\_SECRETS in msgIn.ulExtendedOp and
131. AmILHServer() then
132. /\* Request replication of a single object with secret.
133. \* Secret replication is allowed only if these three conditions
134. \* hold:
135. \* 1. Caller is an RODC. An RODC will always be a member of
136. \* "Enterprise Read-Only Domain Controllers" (RID 498)
137. \* [MS-ADTS] section 6.1.1.6.14.
138. \* 2. The object is configured to reveal secrets.
139. \* 3. Outbound secret replication is not disabled.
140. \*/
141. serverObj := clientDsaObj!parent
142. rodcObj := serverObj!serverReference
143. if CheckGroupMembership(
144. GetCallerAuthorizationInfo(), SidFromStringSid("S-1-5-22"))
145. and RevealSecretsForUserAllowed(rodcObj, fsmoObj)
146. and (not NTDSDSA\_OPT\_DISABLE\_OUTBOUND\_REPL
147. in DSAObj()!options
148. or DRS\_SYNC\_FORCED in msgIn.ulFlags) then
149. scope := {fsmoObj}
150. else
151. scope := {}
152. endif
153. else if EXOP\_REPL\_OBJ in msgIn.ulExtendedOp
154. if AmILHServer() = true and
155. NTDSDSA\_OPT\_DISABLE\_OUTBOUND\_REPL in DSAObj()!options and
156. not DRS\_SYNC\_FORCED in msgIn.ulFlags then
157. /\* replication of single object is disabled \*/
158. pmsgOut.dwDRSError := ERROR\_DS\_DRA\_SOURCE\_DISABLED
159. return
160. endif
161. /\* Operation is invalid if destination is full replica but this server
162. \* is not, or if both are partial replicas but this server does not have
163. \* all the attributes needed by the destination in its PAS. \*/
164. if(not FullReplicaExists(GetObjectNC(msgIn.pNC^)) and
165. not msgIn.pPartialAttrSet = null)
166. msgOut.ulExtendedRet := EXOP\_ERR\_PARAM\_ERR
167. return
168. else if not GetFilteredAttributeSet() ∩ msgIn.pPartialAttrSet = {} then
169. msgOut.ulExtendedRet := EXOP\_ERR\_PARAM\_ERR
170. return
171. endif
172. scope := {fsmoObj}
173. else
174. /\* Unrecognized request. \*/
175. msgOut.ulExtendedRet := EXOP\_ERR\_UNKNOWN\_OP
176. return
177. endif
178. if scope ≠ {} then
179. /\* Add updates in scope to the response. \*/
180. GetChangesInScope(scope, msgIn.pUpToDateVecDest, msgIn.ulExtendedOp,
181. msgIn.pPartialAttrSet, msgIn.pPartialAttrSet,
182. 0, changedObjs, changedLinks)
183. foreach o in changedObjs
184. AddObjToResponse(
185. hDrs, o, GetObjectNC(msgIn.pNC^), msgIn.ulFlags, msgIn.ulExtendedOp, msgOut)
186. endfor
187. foreach v in changedLinks
188. AddLinkToResponse(v, msgIn, msgOut)
189. endfor
190. endif
191. msgOut.ulExtendedRet := EXOP\_ERR\_SUCCESS
192. return

##### RevealSecretsPolicy

1. typedef enum
2. {
3. RevealSecretsDeny = 0,
4. RevealSecretsAllow = 1,
5. RevealSecretsNoPolicy = 2
6. } RevealSecretsPolicy;

##### GetRevealSecretsPolicyForUser

1. procedure GetRevealSecretsPolicyForUser(
2. rodcObj: DSName, userObj: DSName): RevealSecretsPolicy

*Informative summary of behavior*: The GetRevealSecretsPolicyForUser procedure returns the policy that indicates whether the server that holds the secrets of the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) *userObj* is allowed to send those secrets to the [**RODC**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870) identified by the RODC [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *rodcObj*. If the policy explicitly prohibits the RODC from receiving the secrets, RevealSecretsDenied is returned. If the policy explicitly allows the RODC to receive the secrets, RevealSecretsAllow is returned. In all other cases, RevealSecretsNoPolicy is returned.

1. neverRevealObj: DSName
2. revealObj: DSName
3. /\* An RODC can always cache secrets of its own account
4. \*/
5. if rodcObj = userObj /\* see section 5 DSNAME for DSName equality \*/
6. then
7. return RevealSecretsAllow
8. endif
9. /\* An RODC can always cache secrets of its own
10. \* secondary Kerberos TGT account but not other
11. \* secondary Kerberos TGT accounts.
12. \* See [MS-KILE]
13. \*/
14. if rodcObj!msDS-KrbTgtLink = userObj then
15. return RevealSecretsAllow
16. endif
17. krbtgts = select o from children DefaultNC() where
18. o!msDS-KrbTgtLink ≠ null
19. foreach krbtgt in krtgts do
20. if userObj = krbtg!msDS-KrbTgtLink then
21. return RevealSecretsDeny
22. endif
23. endfor
24. /\* Never reveal secrets of inter-domain
25. \* trust accounts
26. \*/
27. if userObj!UserAccountControl ∩ {ADS\_UF\_INTERDOMAIN\_TRUST\_ACCOUNT}
28. ≠ {} then
29. return RevealSecretsDeny
30. endif
31. /\* Never reveal secrets of users reachable from
32. \* rodcObj!msDS-NeverRevealGroup
33. \*/
34. foreach neverRevealObj in rodcObj!msDS-NeverRevealGroup
35. if IsUserIncluded(
36. userObj!objectSid, neverRevealObj!objectSid) then
37. return RevealSecretsDeny
38. endif
39. endfor
40. /\* Only reveal secrets of users reachable from
41. \* rodcObj!msDS-RevealOnDemandGroup
42. \*/
43. foreach revealObj in rodcObj!msDS-RevealOnDemandGroup
44. if IsUserIncluded(
45. userObj!objectSid, revealObj!objectSid) then
46. return RevealSecretsAllow
47. endif
48. endfor
49. return RevealSecretsNoPolicy

##### RevealSecretsForUserAllowed

1. procedure RevealSecretsForUserAllowed(
2. rodcObj: DSName, userObj: DSName): boolean

*Informative summary of behavior*: The RevealSecretsForUserAllowed procedure returns true if a server that holds secrets of the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) *userObj* is allowed to send those secrets to the [**RODC**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870) identified by RODC [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *rodcObj*.

1. policy: RevealSecretsPolicy
2. allowed: boolean
3. policy = GetRevealSecretsPolicyForUser(rodcObj, userObj)
4. if (policy = RevealSecretsDeny) then
5. allowed := false
6. else if (policy = RevealSecretsAllow) then
7. allowed := true
8. else
9. allowed := false
10. endif
11. return allowed

##### GetRoleScope

1. procedure GetRoleScope(fsmoObj: DSName): set of DSName

*Informative summary of behavior*: The GetRoleScope procedure returns the set of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f) identified by the [**FSMO role object**](#gt_6ea17c3e-787a-40e3-a62f-0313dcdc16b7) *fsmoObj*.

1. scope: set of DSName
2. partitionsFsmoObj: DSName
3. schemaFsmoObj: DSName
4. ridFsmoObj: DSName
5. pdcFsmoObj: DSName
6. c: DSName
7. r: set of DSName
8. partitionsFsmoObj := select one o from children ConfigNC()
9. where o!name = "Partitions"
10. schemaFsmoObj := SchemaNC()
11. infrastructureFsmoObj := select one o from children DefaultNC()
12. where o!name = "Infrastructure"
13. ridFsmoObj := DefaultNC()!rIDManagerReference
14. /\* Scope always includes fsmoObj. For the PDC Emulation Role, scope
15. \* includes only fsmoObj. \*/
16. scope := {fsmoObj}
17. if fsmoObj = partitionsFsmoObj then
18. /\* Partition Naming Master Role: Add to scope the children of the
19. \* Partitions container. \*/
20. r := select all o from children partitionsFsmoObj where true
21. scope := scope + r
22. else if fsmoObj = schemaFsmoObj then
23. /\* Schema Master Role: Set scope to all objects in the Schema
24. \* NC. \*/
25. scope := select all o from subtree SchemaNC() where true
26. else if fsmoObj = infrastructureFsmoObj then
27. /\* Infrastructure Master Role: Add to scope all objects in the
28. \* subtree rooted at CN=DomainUpdates,CN=System,DefaultNC(). \*/
29. c := select one o from children DefaultNC() where o!name = "System"
30. c := select one o from children c where o!name = "DomainUpdates"
31. r := select all o from subtree c where true
32. scope := scope + r
33. else if fsmoObj = ridFsmoObj then
34. /\* RID Allocation Master Role: Add to scope all children of
35. \* CN=Infrastructure,DefaultNC() that are of class
36. \* infrastructureUpdate and have a value for the proxiedObjectName
37. \* attribute. \*/
38. r := select all o from children-ts-included infrastructureFsmoObj
39. where infrastructureUpdate in o!objectClass and
40. not o!proxiedObjectName = null
41. scope := scope + r
42. endif
43. return scope

##### SortResponseLinks

1. procedure SortResponseLinks(var msgOut: DRS\_MSG\_GETCHGREPLY\_NATIVE)

The SortResponseLinks procedure sorts the contents of *msgOut.rgValues* in ascending order according to the comparison method CompareLinks():

1. procedure CompareLinks(REPLVALINF\_NATIVE val1, REPLVALINF\_NATIVE val2): integer
2. begin
3. c: integer
4. dsname1: DSName
5. dsname2: DSName
6. /\* Returns 1
7. if val1 > val2, 0 if val1 = val2, or -1 if val1 < val2. \*/
8. /\* Compare by ascending host object objectGUID. \*/
9. c := result of ANSI C function memcmp()
10. applied to val1.pObject^.Guid and val2.pObject^.Guid,
11. in little-endian byte order
12. /\* Then by ascending attribute ID. \*/
13. if c = 0 then
14. if val1.attrTyp < val2.attrTyp then
15. c := -1
16. else if val1.attrTyp > val2.attrType then
17. c := 1
18. endif
19. endif
20. /\* Then by ascending "is present". \*/
21. if c = 0 then
22. if not val1.fIsPresent and val2.fIsPresent then
23. c := -1
24. else if val1.fIsPresent and not val2.fIsPresent then
25. c := 1
26. endif
27. endif
28. /\* Then by ascending referenced object objectGUID. \*/
29. if c = 0 then
30. dsname1 := Value of val1.AVal.pVal^
31. dsname2 := Value of val2.AVal.pVal^
32. c := result of ANSI C function memcmp() applied to dsname1.Guid
33. and dsname2.Guid, in little-endian byte order
34. endif
35. return c
36. end

##### ReplValInfV1ListFromReplValInfNativeList

1. procedure ReplValInfV1ListFromReplValInfNativeList(
2. replValInfNativeList : REPLVALINF\_NATIVE\*) : REPLVALINF\_V1\*

Informative summary of behavior: The ReplValInfV1ListFromReplValInfNativeList procedure transforms a list of [REPLVALINF\_NATIVE](#Section_882a0aa8fb564be6ad4ab9030314111e) structures into a list of [**REPLVALINF\_V1**](#Section_22946fbf170e4ab482c7dabdfd97bf5a) structures. Elements in a native structure that do not exist in a V1 structure are omitted from the V1 structure.

1. returnList : list of REPLVALINF\_V1
2. v1 : REPLVALINF\_V1
3. for each e in replValInfNativeList
4. /\* NOTE: Copy only the fields that exist in a REPLVALINF\_V1 structure. \*/
5. v1 := e
6. add v1 to returnList
7. endfor
8. return returnList

##### ReplValInfNativeListFromReplValInfV1List

1. procedure ReplValInfNativeListFromReplValInfV1List(
2. replValInfV1List : REPLVALINF\_V1\*) : REPLVALINF\_NATIVE\*

Informative summary of behavior: The ReplValInfNativeListFromReplValInfV1List procedure transforms a list of [**REPLVALINF\_V1**](#Section_22946fbf170e4ab482c7dabdfd97bf5a) structures into a list of [REPLVALINF\_NATIVE](#Section_882a0aa8fb564be6ad4ab9030314111e) structures. Elements in a native structure that do not exist in a V1 structure are initialized to 0 or NULL values.

1. returnList : list of REPLVALINF\_NATIVE
2. native : REPLVALINF\_NATIVE
3. for each v1 in replValInfV1List
4. native := 0
5. native := v1
6. add native to returnList
7. endfor
8. return returnList

##### TransformOutput

1. procedure TransformOutput(
2. msgOut: DRS\_MSG\_GETCHGREPLY\_NATIVE,
3. flags: DRS\_OPTIONS,
4. pdwOutVersion: ADDRESS OF DWORD,
5. pmsgOut: ADDRESS OF DRS\_MSG\_GETCHGREPLY): ULONG

*Informative summary of behavior*: The TransformOutput procedure transforms the native reply (a superset of all supported reply messages) into the reply version supported by the client, optionally compressing it. The compression algorithms used for the DRS\_COMP\_ALG\_WIN2K3 algorithm type are specified in section [4.1.10.5.21](#Section_270427c188704691a0463e106cf1e735). The compression algorithm used for the DRS\_COMP\_ALG\_MSZIP algorithm type is specified in [[RFC1951]](https://go.microsoft.com/fwlink/?LinkId=90302).

1. pickled: sequence of BYTE
2. compressed: sequence of BYTE
3. allowedAlgs: set of DRS\_COMP\_ALG\_TYPE
4. compressAlg: DRS\_COMP\_ALG\_TYPE
5. compress: boolean
6. /\* The SMTP transport [MS-SRPL] performs its own compression. \*/
7. compress := DRS\_USE\_COMPRESSION in flags
8. and not DRS\_MAIL\_REP in flags
9. if pdwOutVersion^ < dc.minimumGetChangesReplyVersion
10. return ERROR\_REVISION\_MISMATCH
11. if pdwOutVersion^ = 9 or pdwOutVesion^ = 6 then
12. if pdwOutVersion^ = 9 then
13. /\* Convert to V9.\*/
14. pdwOutVersion^ := 9
15. pmsgOut^.V9 := msgOut
16. else if pdwOutVersion^ = 6 then
17. /\* Convert to V6. \*/
18. pdwOutVersion^ := 6
19. pmsgOut^.V6 := msgOut
20. pmsgOut^.V6.rgValues := ReplValInfV1ListFromReplValInfNativeList(msgOut.rgValues)
21. endif
22. if compress then
23. /\* Note that the only difference between the compression processing of a V2 and \*/
24. /\* a V7 message is that a V7 message can use the DRS\_COMP\_ALG\_WIN2K3 compression \*/
25. /\* algorithm and a V2 message can not. \*/
26. /\* Return V7 (compressed V6 or V9). \*/
27. if not DRS\_EXT\_GETCHGREPLY\_V7 in ext.dwFlags then
28. return ERROR\_REVISION\_MISMATCH
29. endif
30. /\* Serialize msgOut into a byte stream. \*/
31. pickled := Pickling of pmsgOut^.V6 or pmsgOut^.V9, as specified by
32. [C311] Part 2, "IDL/NDR Pickles", and
33. [MS-RPCE] sections 2.2.6 and 2.2.7, "Type Serialization
34. Version 1" and "Type Serialization Version 2"
35. /\* Select a compression algorithm. \*/
36. allowedAlgs := {DRS\_COMP\_ALG\_NONE, DRS\_COMP\_ALG\_MSZIP}
37. if DRS\_EXT\_W2K3\_DEFLATE in ext.dwFlags then
38. allowedAlgs := allowedAlgs + {DRS\_COMP\_ALG\_WIN2K3}
39. endif
40. compressAlg := One of allowedAlgs, selected by an
41. implementation-defined policy.
42. /\* Compress the serialized msgOut. \*/
43. if (compressionAlg = DRS\_COMP\_ALG\_MSZIP) then
44. compressed :=
45. Compress pickled in accordance
46. with [RFC1951].
47. else
48. CompressOrDecompressWin2k3(pickled, compressed, pickled.length, TRUE)
49. endif
50. pmsgOut^.V7.dwCompressedVersion := pdwOutVersion^
51. pmsgOut^.V7.CompressionAlg := compressAlg
52. pmsgOut^.V7.CompressedAny.cbUncompressedSize := pickled.length
53. pmsgOut^.V7.CompressedAny.cbCompressedSize := compressed.length
54. pmsgOut^.V7.CompressedAny.pbCompressedData := bytes in compressed
55. pdwOutVersion^ := 7
56. endif
57. else
58. /\* Return V1 (uncompressed) or V2 (compressed V1). \*/
59. /\* First, convert to V1. \*/
60. pdwOutVersion^ := 1
61. pmsgOut^.V1 := msgOut
62. pmsgOut^.V1.pUpToDateVecSrc := Convert msgOut.pUpToDateVecSrc (of
63. type UPTODATE\_VECTOR\_V1\_EXT) to UPTODATE\_VECTOR\_V2\_EXT by
64. creating a new UPTODATE\_VECTOR\_V1\_EXT with a V1 cursor for each
65. V2 cursor, sans the timeLastSyncSuccess field.
66. /\* V1 has the NC size in the ulExtendedRet field. \*/
67. if msgOut.cNumNcSizeObjects > 0 then
68. pmsgOut^.V1.ulExtendedRet := msgOut.cNumNcSizeObjects
69. endif
70. if compress then
71. /\* Serialize msgOut into a byte stream. \*/
72. pickled := Pickling of pmsgOut^.V1, as specified by
73. [C311] Part 2, "IDL/NDR Pickles" and
74. [MS-RPCE] sections 2.2.6 and 2.2.7, "Type Serialization
75. Version 1" and "Type Serialization Version 2"
76. /\* Select a compression algorithm. \*/
77. allowedAlgs := {DRS\_COMP\_ALG\_NONE, DRS\_COMP\_ALG\_MSZIP}
78. compressAlg := One of allowedAlgs, selected by an
79. implementation-defined policy.
80. /\* Compress the serialized msgOut. \*/
81. if (compressionAlg = DRS\_COMP\_ALG\_MSZIP) then
82. compressed :=
83. Compress pickled in accordance
84. with [RFC1951].
85. else
86. CompressOrDecompressWin2k3(pickled, compressed, pickled.length, TRUE)
87. endif
88. pdwOutVersion^ := 2
89. pmsgOut^.V2.CompressedV1.cbUncompressedSize := pickled.length
90. pmsgOut^.V2.CompressedV1.cbCompressedSize := compressed.length
91. pmsgOut^.V2.CompressedV1.pbCompressedData := bytes in compressed
92. endif
93. endif
94. return 0

##### CompressOrDecompressWin2k3

1. procedure CompressOrDecompressWin2k3(
2. inputBuffer: sequence of BYTE,
3. inputSize: DWORD
4. ref outputBuffer: sequence of BYTE,
5. comp: BOOLEAN)

*Informative summary of behavior*: The CompressOrDecompressWin2k3 procedure compresses or decompresses data using the compression algorithm LZ77 (section [4.1.10.5.21.1](#Section_2ea1ab2599b4489ba71c61e4c196b561)) and the basic encoding algorithm DIRECT2 (section [4.1.10.5.21.2](#Section_0781595613754148b33e39c31f988ec8)). The procedure has the following parameters:

* *inputBuffer*: A sequence of [BYTE](#Section_545826e419454580961f0f0c0a47e797) containing data to compress or decompress.
* *inputSize*: The [DWORD](#Section_60c3f5f194924d1083c89a155e162ef3) value that indicates the size of *inputBuffer* in bytes.
* *outputBuffer*: A sequence of BYTE that is an empty buffer. Compressed or decompressed data is filled into this buffer.
* *comp*: A Boolean indicating whether to compress (*comp*=TRUE) or decompress (*comp*=FALSE) the *inputBuffer*.

###### LZ77 Compression Algorithm

The LZ77 compression algorithm is used to analyze input data and determine how to reduce the size of that input data by replacing redundant information with metadata. Sections of the data that are identical to sections of the data that have been encoded are replaced by a small amount of metadata that indicates how to expand those sections again. The encoding algorithm is used to take that combination of data and metadata and serialize it into a stream of bytes that can later be decoded and decompressed.

**Compression Algorithm Terminology**

The following terms are associated with the compression algorithm. Some of the terms also apply to the DIRECT2 encoding algorithm defined in the next section.

*input stream*: The sequence of bytes to be compressed.

*byte*: The basic data element in the input stream.

*coding position*: The position of the byte in the input stream that is currently being coded (the beginning of the lookahead buffer).

*lookahead buffer*: The byte sequence from the coding position to the end of the input stream.

*window*: A buffer that indicates the number of bytes from the coding position backward. A window of size W contains the last W processed bytes.

*pointer*: Information about the beginning of the match in the window (referred to as "B" in the example later in this section) and its length (referred to as "L" in the example later in this section).

*match*: The string that is used to find a match of the byte sequence between the lookahead buffer and the window.

**Using the Compression Algorithm**

To use the LZ77 compression algorithm:

1. Set the coding position to the beginning of the input stream.
2. Find the longest match in the window for the lookahead buffer.
3. Output the P,C pair, where P is the pointer to the match in the window, and C is the first byte in the lookahead buffer that does not match.
4. If the lookahead buffer is not empty, move the coding position (and the window) L+1 bytes forward.
5. Return to step 2.

**Compression Process**

The compression algorithm searches the window for the longest match with the beginning of the lookahead buffer and then outputs a pointer to that match. Because even a 1-byte match might not be found, the output cannot contain only pointers. The compression algorithm solves this problem by outputting after the pointer the first byte in the lookahead buffer after the match. If no match is found, the algorithm outputs a null-pointer and the byte at the coding position.

**Compression Process Example**

The following table shows the input stream that is used for this compression example. The bytes in the input, "AABCBBABC", occupy the first nine positions of the stream.

**Input stream**

1. Position 1 2 3 4 5 6 7 8 9
2. Byte A A B C B B A B C

The following table shows the output from the compression process. The table includes the following columns:

Step: Indicates the number of the encoding step. A step in the table finishes every time that the encoding algorithm makes an output. With the compression algorithm, this process happens in each pass through step 3.

Position: Indicates the coding position. The first byte in the input stream has the coding position 1.

Match: Shows the longest match found in the window.

Byte: Shows the first byte in the lookahead buffer after the match.

Output: Presents the output in the format (B,L)C, where (B,L) is the pointer (P) to the match. This gives the following instructions to the decoder: Go back B bytes in the window and copy L bytes to the output. C is the explicit byte.

**Note**  One or more pointers might be included before the explicit byte that is shown in the Byte column. That is, a metadata pointer does not always need to be followed by an explicit byte. An input stream of "ABCABCABC", for example, can be represented as "(0,0)A(0,0)B(0,0)C(3,3)(6,3)" using the (B,L)C notation, with the last two elements being pointers without explicit bytes. The compressed output can be any combination of pointers and explicit bytes.

**Compression process output**

| Step | Position | Match | Byte | Output |
| --- | --- | --- | --- | --- |
| 1. | 1 | -- | A | (0,0)A |
| 2. | 2 | A | B | (1,1)B |
| 3. | 4 | -- | C | (0,0)C |
| 4. | 5 | B | B | (2,1)B |
| 5. | 7 | A B | C | (5,2)C |

The result of compression, conceptually, is the output column—that is, a series of bytes and optional metadata that indicates whether that byte is preceded by some sequence of bytes that is already in the output.

Because representing the metadata itself requires bytes in the output stream, it is inefficient to represent a single byte that has previously been encoded by two bytes of metadata (offset and length). The overhead of the metadata bytes equals or exceeds the cost of outputting the bytes directly. Therefore, the protocol considers sequences of bytes to be a match only if the sequences have three or more bytes in common.

###### DIRECT2 Encoding Algorithm

The basic notion of the DIRECT2 encoding algorithm is that data appears unchanged in the compressed representation, and metadata is encoded in the same output stream, and in line with, the data.

The key to decoding the compressed data is recognizing what bytes are metadata and what bytes are data. The decoder MUST be able to identify the presence of metadata in the compressed and encoded data stream. Bitmasks are inserted periodically in the byte stream to provide this information to the decoder.

This section describes the bitmasks that enable the decoder to distinguish data from metadata. It also describes the process of encoding the metadata.

**Bitmask**

To distinguish data from metadata in the compressed byte stream, the data stream begins with a 4-byte bitmask that indicates to the decoder whether the next byte to be processed is data (a "0" value in the bit), or if the next byte (or series of bytes) is metadata (a "1" value in the bit). If a "0" bit is encountered, the next byte in the input stream is the next byte in the output stream. If a "1" bit is encountered, the next byte or series of bytes is metadata that MUST be interpreted further.

For example, a bitmask of 0x01000000 indicates that the first seven bytes are actual data, followed by encoded metadata that starts at the eighth byte. The metadata is followed by 24 additional bytes of data. A bitmask of 0x112000000 indicates that there will be metadata in the 4th, 8th, and 11th elements (note that the actual byte positions in the compressed data might be different because metadata elements will range from 2 to 6 bytes in length), with the remaining elements being data bytes.

When the bitmask has been consumed, the next four bytes in the input stream are another bitmask.

The bitmask must also contain a "1" in the bit following the last encoded element, to indicate the end of the compressed data. For example, given a hypothetical 8-bit bitmask, the string "ABCABCDEF" is compressed as (0,0)A(0,0)B(0,0)C(3,3)D(0,0)E(0,0)F. Its bitmask would be b'00010001' (0x11). This would indicate three bytes of data, followed by metadata, followed by an additional 3 bytes, finally terminated with a "1" to indicate the end of the stream.

The final end bit is always necessary, even if an additional bitmask has to be allocated. If the string in the above example was "ABCABCDEFG", for example, it would require an additional bitmask. It would begin with the bitmask b'00010000', followed by the compressed data, and followed by another bitmask with a "1" as the next bit to indicate the end of the stream.

**Encoding Metadata**

In the output stream, actual data bytes are stored unchanged. Bitmasks are stored periodically to indicate whether the next byte or bytes are data or metadata. If the next bit in the bitmask is a "1", the next set of bytes in the input data stream is metadata (unless the last element of data was read, in which case the "1" bit would indicate the end of the stream as noted above). This metadata contains an offset back to the start of the data to be copied to the output stream, and the length of the data to be copied.

To represent the metadata as efficiently as possible, the encoding of that metadata is not fixed in length. The encoding algorithm supports the largest possible floating compression window to increase the probability of finding a large match; the larger the window, the greater the number of bytes that are needed for the offset. The encoding algorithm also supports the longest possible match; the longer the match length, the greater the number of bytes that are needed to encode the length.

**Metadata Offset**

The protocol assumes the metadata is two bytes in length. The three low-order bits are used to encode the length. The high-order 13 bits are a first complement of the offset, which is represented as a negative signed value in 2's complement. The offset is only encoded with those 13 bits. This value cannot be extended and defines the maximum size of the compression floating window. For example, the metadata 0x0018 is converted into the offset b'000000000011', and the length b'000'. The offset is '-4', computed by inverting the offset bits, treating the result as a 2's complement, and converting it to an integer.

**Match Length**

Unlike the metadata offset, the match length is extensible. If the length is less than 10 bytes, it is encoded in the three low-order bits of the 2-byte metadata. Although three bits seems to allow for a maximum length of six (the value b'111' is reserved), because the minimum match is three bytes, these three bits actually allow for the expression of lengths from three to nine. The match length goes from L = b'000' + 3 bytes, to L = b'110' + 3 bytes. Because smaller lengths are much more common than the larger lengths, the algorithm tries to optimize for smaller lengths. To encode a length between three and nine, we use the three bits that are "in-line" in the 2-byte metadata.

If the length of the match is greater than nine bytes, an initial bit pattern of b'111' is put in the three bits. This does not signify a length of 10 bytes, but instead a length that is greater than or equal to 10, which is included in the low-order nibble of the following byte.

Every other time that the length is greater than nine, an additional byte follows the initial 2-byte metadata. The first time that the additional byte is included, the low-order nibble is used as the additive length. The high-order nibble is "reserved" for the next metadata instance when the length is greater than nine. Therefore, the first time that the decoder encounters a length that is greater than nine, it reads the next byte from the data stream and the low-order nibble is extracted and used to compute the length for this metadata instance. The high-order nibble is remembered and used the next time that the decoder encounters a metadata length that is greater than nine. The third time that a length that is greater than nine is encountered, another extra byte is added after the 2-byte metadata, with the low-order nibble used for this length and the high-order nibble reserved for the fourth length that is greater than nine, and so on.

If the nibble from this "shared" byte is all "1s" (for example, b'1111'), another byte is added after the shared byte to hold more length. In this manner, a length of 24 is encoded as follows:

* b'111' (in the three bits in the original two bytes of metadata), plus
* b'1110' (in the nibble of the "shared' byte" of extended length)
* b'111' means 10 bytes plus b'1110', which is 14, which results in a total of 24.

If the length is more than 24, the next byte is also used in the length calculation. In this manner, a length of 25 is encoded as follows:

* b'111' (in the three bits in the original two bytes of metadata), plus
* b'1111' (in the nibble of the "shared" byte of extended length), plus
* b'00000000' (in the next byte).

This scheme is good for lengths of up to 278 (a length of 10 in the three bits in the original two bytes of metadata, plus a length of 15 in the nibble of the "shared" byte of extended length, plus a length of up to 254 in the extra byte).

A "full" (all b'1') bit pattern (b'111', b'1111', and b'11111111') means that there is more length in the following two bytes.

The final two bytes of length differ from the length information that comes earlier in the metadata. For lengths that are equal to 280 or greater, the length is calculated only from these last two bytes, and is not added to the previous length bits. The value in the last two bytes, a 16-bit integer, is three less than the metadata length. These last two bytes allow for a match length of up to 32,768 bytes + 3 bytes (the minimum match length).

The following table summarizes the length representation in metadata.

**Note**  Length is computed from the bits that are included in the metadata plus the minimum match length of three.

**Length representation in metadata**

| Match length | Length bits in the metadata |
| --- | --- |
| 24 | b'111' (three bits in the original two bytes of metadata)  +  b'1110' (in the high-order or lower-order nibble, as appropriate, of the shared byte) |
| 25 | b'111' (three bits in the original two bytes of metadata)  +  b'1111' (in the high-order or lower-order nibble, as appropriate, of the shared byte)  +  b'00000000' (in the next byte) |
| 26 | b'111' (three bits in the original two bytes of metadata)  +  b'1111' (in the high-order or lower-order nibble, as appropriate, of the shared byte)  +  b'00000001' (in the next byte) |
| 279 | b'111' (three bits in the original two bytes of metadata)  +  b'1111' (in the high-order or lower-order nibble, as appropriate, of the shared byte)  +  b'11111110' (in the next byte) |
| 280 | b'111' (three bits in the original two bytes of metadata)  b'1111' (in the high-order or lower-order nibble, as appropriate, of the shared byte)  b'11111111' (in the next byte)  0x0115 (in the next two bytes). These two bytes represent a length of 277 + 3 (minimum match length).  **Note**  All of the length is included in the final two bytes and is not additive, as were the previous length calculations for lengths that are smaller than 280 bytes. |
| 281 | b'111' (three bits in the original two bytes of metadata)  b'1111' (in the high-order or lower-order nibble, as appropriate, of the shared byte)  b'11111111' (in the next byte)  0x0116 (in the next two bytes). This is 278 + 3 (minimum match length).  **Note**  All of the length is included in the final two bytes and is not additive, as were the previous length calculations for lengths that are smaller than 280 bytes. |

A "full" bit pattern in that last half word does not mean that more metadata is coming after the last bytes.

The LZ77 compression algorithm produces a well-compressed encoding for small valued lengths, but as the length increases, the encoding becomes less well compressed. A match length of greater than 278 bytes requires a relatively large number of bits: 3+4+8+16. This includes three bits in the original two bytes of metadata, four bits in the nibble in the "shared" byte, eight bits in the next byte, and 16 bits in the final two bytes of metadata.

##### GetOptionalFeatureBit

1. procedure GetOptionalFeatureBit(featureGuid: GUID, var bit: integer): boolean

*Informative summary of behavior*: The GetOptionalFeatureBit procedure obtains the *bit* number in the dwFlagsExt and dwExtCaps fields of the DRS\_EXTENSIONS\_INT structure that corresponds to the [**optional feature**](#gt_785b66f1-22b3-450f-97aa-a24a39d04d47) identified by *featureGuid*.

1. if (featureGUID = GUID of Recycle Bin optional feature)
2. /\* [MS-ADTS] section 6.1.1.2.4.1.3.1 \*/
3. bit := DRS\_EXT\_RECYCLE\_BIN
4. return true
5. else
6. return false
7. endif

#### Client Behavior When Receiving the IDL\_DRSGetNCChanges Response

The client processes an [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) response in relation to the current state of its [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) as detailed in [ProcessGetNCChangesReply](#Section_cb8c38148c044de9b55369db68aeee58) below. This processing, though sometimes complex, is critical to ensuring that each NC replica arrives at the same abstract state.

##### ProcessGetNCChangesReply

1. procedure ProcessGetNCChangesReply(
2. hDrs: DRS\_HANDLE,
3. rf: RepsFrom,
4. msgIn: DRS\_MSG\_GETCHGREQ\_V10,
5. dwOutVersion: ULONG,
6. msgOut: DRS\_MSG\_GETCHGREPLY) : ULONG

*Informative summary of behavior*: The ProcessGetNCChangesReply procedure is invoked when an [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) response is received over [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) or SMTP, as specified in [[MS-SRPL]](%5bMS-SRPL%5d.pdf#Section_ec69eea50d5e428ab5bc66732aaeb866). Processing of a given response can be separated into five distinct phases: decompression, [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) value decryption, processing [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493), processing [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) updates, and updating the "watermark" information.

The arguments to this procedure are as follows:

* *hDrs*: The [DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) derived by sending [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) to the server.
* *rf*: [RepsFrom](#Section_3ef27d3cb9c944048e53ebf3a64a9a10) for the server.
* *msgIn*: IDL\_DRSGetNCChanges request message sent to the server.
* *dwOutVersion*: Version of response message received from the server.
* *msgOut*: Response message received from the server.

1. msgReplyNative: DRS\_MSG\_GETCHGREPLY\_NATIVE
2. replEntinfList: REPLENTINFLIST
3. continueProcessing: boolean
4. writableReplica: boolean
5. sourcePrefixTable: PrefixTable
6. attributesAndStamps: set of AttributeAndStamp
7. linkValueCount: DWORD
8. clientSchemaSignature: sequence of BYTE
9. serverSchemaSignature: sequence of BYTE
10. fServerSchemaMoreRecent: boolean
11. lastElement: DWORD
12. ulResult : ULONG
13. /\* Decompress and/or translate the response to a Native response,
14. \* as necessary. \*/
15. if (dwOutVersion = 0x2) or (dwOutVersion = 0x7) then
16. msgReplyNative := DecompressReplyMessage(msgOut, dwOutVersion)
17. else
18. msgReplyNative := GetNCChangesNativeReply(msgOut, dwOutVersion)
19. endif
20. ulResult := msgReplyNative.dwDRSError
21. if (ulResult = 0) then
22. sourcePrefixTable :=
23. AbstractPTFromConcretePT(msgReplyNative.PrefixTableSrc)
24. /\* Check whether the schema on client and server match. \*/
25. lastElement := sourcePrefixTable.length - 1
26. serverSchemaSignature :=
27. copy sourcePrefixTable[lastElement].prefix.length bytes of data
28. from sourcePrefixTable[lastElement].prefix.elements
29. clientSchemaSignature := SchemaInfo()
30. if clientSchemaSignature ≠ serverSchemaSignature and
31. msgReplyNative.pNC^ ≠ SchemaNC())
32. then
33. return ERROR\_DS\_DRA\_SCHEMA\_MISMATCH
34. endif
35. Remove sourcePrefixTable[lastElement] from sourcePrefixTable
36. else
37. return ulResult
38. endif
39. /\* If the source has the Recycle Bin optional feature enabled, then it must
40. \* be enabled locally, unless the Schema partition is being updating.
41. \*/
42. if msgReplyNative.pNC^ ≠ SchemaNC() and
43. ServerExtensions(hDrs).RB and not IsRecycleBinEnabled()
44. then
45. EnableRecycleBin()
46. return ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED
47. endif
48. /\* If the source has the Privileged Access Management optional feature enabled,
49. \* then it must be enabled locally unless the Schema partition is being updated.
50. \*/
51. if msgReplyNative.pNC^ ≠ SchemaNC() and
52. ServerExtensions(hDrs).GR9 and not IsPrivilegedAccessManagementEnabled()
53. then
54. EnablePrivilegedAccessManagement()
55. return ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED
56. endif
57. /\* Process object updates. \*/
58. replEntinfList := msgReplyNative.pObjects^
59. while (ulResult = 0) and (not replEntinfList = null)
60. /\* Decrypt any encrypted attribute values. \*/
61. ulResult := DecryptValuesIfNecessary (
62. hDrs,
63. sourcePrefixTable,
64. replEntinfList.Entinf.AttrBlock)
65. if (ulResult = 0) then
66. attributesAndStamps := GetStampsForUpdate(
67. replEntinfList,
68. sourcePrefixTable)
69. /\* Process objects that are moved across an NC. \*/
70. continueProcessing := PrepareCrossNCMove(
71. replEntinfList,
72. sourcePrefixTable)
73. endif
74. if continueProcessing and (ulResult = 0) then
75. if (DRS\_WRIT\_REP in msgIn.ulFlags) then
76. writableReplica := true
77. else
78. writableReplica := false
79. endif
80. continueProcessing := AdjustInstanceTypeAttrVal(
81. msgReplyNative.pNC^,
82. writableReplica ,
83. replEntinfList,
84. prefixTable)
85. endif
86. if continueProcessing and (ulResult = 0) then
87. if (not ObjExists(replEntinfList.Entinf.pName^)) then
88. ulResult := AddObject(
89. replEntinfList,
90. sourcePrefixTable,
91. attributesAndStamps)
92. else
93. ulResult := UpdateObject(
94. replEntinfList,
95. sourcePrefixTable,
96. attributesAndStamps)
97. endif
98. endif
99. replEntinfList := replEntinfList.pNextEntInf^
100. endwhile
101. /\* Enable link value updates for outbound replication
102. \* if inbound link value updates are detected from source. \*/
103. if (msgReplyNative.cNumValues > 0) then
104. dc.fLinkValueStampEnabled = true
105. endif
106. /\* Process link value updates. \*/
107. linkValueCount := 0
108. while (ulResult = 0) and (linkValueCount < msgReplyNative.cNumValues)
109. ulResult := ProcessLinkValue(
110. msgReplyNative.rgValues[linkValueCount],
111. msgReplyNative.pNC^,
112. prefixTable,
113. msgIn.ulFlags,
114. msgIn.ulMoreFlags)
115. linkValueCount := linkValueCount + 1
116. endwhile
117. if (ulResult = ERROR\_DS\_DRA\_MISSING\_PARENT) then
118. Send IDL\_DRSGetNCChanges message again with the same input
119. parameters specified in msgIn but this time with msgIn.ulFlags
120. containing DRS\_GET\_ANC field set. It is an error for this
121. condition to occur if (DRS\_GET\_ANC in msgIn.ulFlags) is true
122. else if (ulResult = ERROR\_DS\_DRA\_RECYCLED\_TARGET) then
123. Send IDL\_DRSGetNCChanges message again with the same input
124. parameters specified in the msgIn but this time with msgIn.ulMoreFlags
125. containing DRS\_GET\_TGT field set.
126. else if (msgIn.ulExtendedOp = 0) then
127. /\* Not an extended operation. Update "watermark" information. \*/
128. UpdateRepsFrom(
129. rf,
130. msgReplyNative,
131. dsaServer,
132. ulResult)
134. if (ulResult = 0) and (msgReplyNative.fMoreData = false) then
135. UpdateUTDandPAS(
136. msgReplyNative,
137. msgIn.partialAttrSetEx^)
138. endif
139. endif
140. return ulResult

##### EnableRecycleBin

1. procedure EnableRecycleBin()

*Informative summary of behavior*: The EnableRecycleBin procedure is invoked during inbound [replication](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) if the source has the [Recycle Bin](#gt_54624800-58f4-45e9-90bf-c9b52dcf98f3) [optional feature](#gt_785b66f1-22b3-450f-97aa-a24a39d04d47) enabled but the destination does not. It adds a reference to the [object](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) representing the Recycle Bin optional feature to the msDS-EnabledFeature [attribute](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of both the nTDSDSA object of the destination [DC](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) and the Cross-Ref-Container [container](#gt_c3143e71-2ada-417e-83f4-3ef10eff2c56). For more details, see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) sections 3.1.1.9, 3.1.1.9.1, and 6.1.1.2.1.

1. Asynchronous Processing: Initiate a logical thread of control
2. to process the remainder of this request asynchronously
3. and then return.
4. DSNAME rbObj := select one v from ConfigNC()
5. where v!objectGuid = 766ddcd8-acd0-445e-f3b9-a7f9b6744f2a
6. ConfigNC()!msDS-EnabledFeature :=
7. ConfigNC()!msDS-EnabledFeature + {rbObj}
8. DSAObj()!msDS-EnabledFeature :=
9. DSAObj()!msDS-EnabledFeature + {rbObj}
10. return

##### EnablePrivilegedAccessManagement

1. procedure EnablePrivilegedAccessManagement()

Informative summary of behavior: The EnablePrivilegedAccessManagement procedure is invoked during inbound [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) if the source has the [**Privileged Access Management**](#gt_2cc7e2f1-0f12-4357-9846-e772aef4de39) [**optional feature**](#gt_785b66f1-22b3-450f-97aa-a24a39d04d47) enabled but the destination does not. It adds a reference to the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) representing the Privileged Access Management optional feature to the msDS-EnabledFeature [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of both the [**nTDSDSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8) of the destination [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) and the Cross-Ref-Container [**container**](#gt_c3143e71-2ada-417e-83f4-3ef10eff2c56). For more details, see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) sections 3.1.1.9, 3.1.1.9.2, and 6.1.1.2.1.

1. Asynchronous Processing: Initiate a logical thread of control
2. to process the remainder of this request asynchronously
3. and then return
4. DSNAME elObj := select one v from ConfigNC()
5. where v!objectGuid = ec43e873-cce8-4640-b4ab-07ffe4ab5bcd
6. ConfigNC()!msDS-EnabledFeature :=
7. ConfigNC()!msDS-EnabledFeature + {elObj}
8. DSAObj()!msDS-EnabledFeature :=
9. DSAObj()!msDS-EnabledFeature + {elObj}
10. return

##### PrepareCrossNCMove

1. procedure PrepareCrossNCMove(
2. replEntinfList: REPLENTINFLIST,
3. sourcePrefixTable: PrefixTable): boolean

*Informative summary of behavior*: The PrepareCrossNCMove procedure determines whether the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) specified by the *replEntinfList* argument is being moved from one [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) to another and, if so, performs preparatory work and/or terminates further processing of *replEntinfList*. The procedure returns true if further processing of [**replicated update**](#gt_2a923099-db0a-4932-af28-4354601e85c4) in *replEntinfList* has to be performed. Otherwise, it returns false.

1. proxiedNameAttrVal: ATTRVAL
2. proxiedNameValue: DNBinary
3. localProxiedNameValue: DNBinary
4. proxyEpoch: DWORD
5. localProxyEpoch: DWORD
6. proxyObject: DSName
7. proxyObjectNameValue: DNBinary
8. isProxy: boolean
9. objClassVal: ATTRVAL
10. proxiedNameAttrVal := ENTINF\_GetValue(
11. replEntinfList.Entinf,
12. proxiedObjectName,
13. sourcePrefixTable)
14. if (proxiedNameAttrVal = null) then
15. /\* Update is not related to cross NC move. Therefore, continue
16. processing the replicated update. \*/
17. return true
18. endif
19. /\* replEntinfList corresponds to an object that has moved across an
20. \* NC /
21. proxiedNameValue := ValueFromATTRVAL(
22. proxiedNameAttrVal, Syntax(proxiedObjectName), sourcePrefixTable)
23. proxyEpoch := GetProxyEpoch(proxiedNameValue)
24. /\* Check whether the objectClass is infrastructureUpdate. \*/
25. objClassVal := ENTINF\_GetValue(replEntinfList.Entinf, objectClass,
26. sourcePrefixTable)
27. if LocalAttidFromRemoteAttid(
28. sourcePrefixTable, objClassVal.pAVal^.pVal^)
29. = infrastructureUpdate then
30. isProxy := true
31. else
32. isProxy := false
33. endif
34. if not isProxy then
35. /\* Replicated update is not for an infrastructureUpdate object. \*/
36. proxyObject := replEntinfList.Entinf.pName^
37. if (ObjExists(proxyObject)) and
38. (not proxyObject!proxiedObjectName = null) then
39. localProxyEpoch := GetProxyEpoch(proxyObject!proxiedObjectName)
40. else
41. localProxyEpoch := 0
42. endif
43. if (localProxyEpoch > proxyEpoch) then
44. /\* Local EPOCH value is higher. Don't continue processing the
45. \* replicated update. \*/
46. return false
47. else if (localProxyEpoch < proxyEpoch) and
48. (ObjExists(proxyObject)) then
49. Expunge(proxyObject)
50. endif
51. else
52. proxyObjectNameValue :=
53. ValueFromATTRVAL(proxiedNameAttrVal.pVal,
54. Syntax(proxiedObjectName),
55. sourcePrefixTable)
56. proxyObject := proxyObjectNameValue.dn
57. if (ObjExists(proxyObject)) then
58. localProxiedNameValue = proxyObject!proxiedObjectName
59. if (localProxiedNameValue = null) then
60. localProxyEpoch := 0
61. else
62. localProxyEpoch := GetProxyEpoch(localProxiedNameValue)
63. endif
64. if (localProxyEpoch < proxyEpoch) then
65. Expunge(proxyObject)
66. endif
67. endif
68. endif
69. return true /\* Continue processing the replicated update. \*/

##### AdjustInstanceTypeAttrVal

1. procedure AdjustInstanceTypeAttrVal(
2. ncReplicated: DSName,
3. writableReplica: DSName,
4. var replEntinfList: REPLENTINFLIST,
5. prefixTable: PrefixTable) : boolean

*Informative summary of behavior*: The AdjustInstanceTypeAttrVal procedure adjusts the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) value of instanceType attribute in *replEntinfList* parameter to an appropriate value that suits the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) on the client. The procedure returns true if further processing of [**replicated update**](#gt_2a923099-db0a-4932-af28-4354601e85c4) in *replEntinfList* has to be performed. Otherwise, it returns false.

1. instanceTypeAttrVal: ATTRVAL
2. instanceTypeAdjustedAttrVal: ATTRVAL
3. instanceTypeVal: ULONG
4. instanceTypeAdjustedVal: ULONG
5. ncSubRef: DSName
6. instanceTypeAttrVal := ENTINF\_GetValue(replEntinfList.Entinf,
7. instanceType, prefixTable)
8. if (instanceTypeAttrVal = null) then
9. /\* If instanceType attribute is not present in Entinf
10. \* then there is no value to adjust. \*/
11. return true
12. endif
13. instanceTypeVal := ValueFromATTRVAL(
14. instanceTypeAttrVal, Syntax(instanceType), prefixTable)
15. if (IT\_NC\_HEAD in instanceTypeVal) and
16. (not ncReplicated = replEntinfList.Entinf.pName^)
17. /\* If IT\_NC\_HEAD is set in instanceTypeVal and
18. \* replEntinfList.Entinf.pName is not the DSName of the root of the
19. \* NC replica that the client is replicating, then this object is
20. \* a subordinate reference. Take this opportunity
21. \* to ensure that ncReplicated!subRefs has an entry for this
22. \* sub-ref object.
23. \*/
24. ncSubRef := replEntinfList.Entinf.pName^
25. if (not ncSubRef in ncReplicated!subRefs) then
26. ncReplicated!subRefs := ncReplicated!subRefs + {ncSubRef}
27. endif
28. if ObjExists(ncSubRef)
29. /\* Ensure that all sub-ref objects have the flag IT\_NC\_ABOVE set. \*/
30. if not IT\_NC\_ABOVE in ncSubRef!instanceType then
31. ncSubRef!instanceType := ncSubRef!instanceType + {IT\_NC\_ABOVE}
32. endif
33. /\* If the sub-ref object corresponds to a locally instantiated
34. \* child NC, then skip this update as the sub-ref object
35. \* will be updated when the child NC replicates in.
36. \*/
37. if PartialGCReplicaExists(ncSubRef) then
38. return false /\* Skip processing this entry. \*/
39. endif
40. endif
41. /\* If sub-ref object does not exist or exists but the child NC is not
42. \* locally instantiated, then continue processing this entry to
43. \* add or update the sub-ref object.
44. \*/
45. instanceTypeAdjustedVal := instanceTypeVal + {IT\_NC\_ABOVE,
46. IT\_UNINSTANT, IT\_NC\_HEAD}
47. else
48. if (not writableReplica) and
49. (IT\_WRITE in instanceTypeVal) then
50. /\* If the client NC replica is a partial replica then remove the
51. \* IT\_WRITE flag from the instanceTypeVal to mark the object as
52. \* read-only.
53. \*/
54. instanceTypeAdjustedVal := instanceTypeVal - {IT\_WRITE}
55. else
56. instanceTypeAdjustedVal := instanceTypeVal
57. endif
58. endif
59. /\* Set or reset instance type bits other than IT\_WRITE and
60. \* IT\_NC\_HEAD. \*/
61. instanceTypeAdjustedVal :=
62. SetResetInstanceTypeBits(instanceTypeAdjustedVal)
63. instanceTypeAdjustedAttrVal := ATTRVALFromValue(
64. instanceTypeAdjusted, Syntax(instanceType), prefixTable)
65. ENTINF\_SetValue(replEntinfList.Entinf, instanceType,
66. instanceTypeAdjustedAttrVal, prefixTable)
67. return true

##### SetResetInstanceTypeBits

1. procedure SetResetInstanceTypeBits(y: DWORD): DWORD

The SetResetInstanceTypeBits procedure is an implementation-specific function that MAY[<31>](#Appendix_A_31" \o "Product behavior note 31) set or reset bits in *y* other than IT\_WRITE and IT\_NC\_HEAD. It returns the [**updated**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) value.

##### PerformModifyOperation

1. procedure PerformModifyOperation(
2. data: ENTINF,
3. updateObject: DSNAME,
4. prefixTable: PrefixTable): integer

The PerformModifyOperation procedure performs a modify operation with the given [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b) to modify *updateObject*, an existing [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9). For more details, see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.5.3.

This operation modifies the object whose [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) is *updateObject*. If the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) in data.pName.StringName is not equal to the DN of *updateObject* in updateObject.StringName, then let *newParentName* be the DSNAME of the [**parent object**](#gt_0d41951a-62f0-4fbd-bb23-22f645ae3bf5) identified in data.pName.StringName, let *newRDN* be the [**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) identified in data.pName.StringName, and call PerformModifyDNOperation(*updateObject!distinguishedName*, *newParentName*, *newRDN*). The PerformModifyDNOperation procedure call is NOT to be performed as an [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20).

For each [ATTR](#Section_a2db41e278034d3ca4990fee92b1c149) **attr** in data.AttrBlock, let *attribute* be the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) returned by [LocalAttidFromRemoteAttid](#Section_5e30bd01b1dd4019b4061e9c2472f359)(*prefixTable*, attr.attrType). Then on the object modified by PerformModifyOperation, if the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) whose ATTRTYP is *attribute* is present, all previous values are removed and replaced with the values attr.AttrVal.pAVal[0... attr.AttrVal.valCount]. If the attribute whose ATTRTYP is *attribute* is not present, it is added with the values attr.AttrVal.pAVal[0... attr.AttrVal.valCount].

The PerformModifyOperation procedure is NOT to be performed as an originating update. The AttributeAndStamp values associated with the modifed attributes must not be touched by this procedure. For more details about originating updates, see [MS-ADTS] section 3.1.1.1.9.

If the modify operation succeeds, the procedure returns 0. If the modify operation fails, the procedure returns a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

##### NameObject

1. procedure NameObject(
2. replEntinfList: REPLENTINFLIST,
3. sourcePrefixTable: PrefixTable,
4. nc: DSName,
5. attributesAndStamps: set of AttributeAndStamp): DWORD

*Informative summary of behavior*: The NameObject procedure performs the necessary steps to identify an unused name for an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca), whether it is being added new or renamed. This procedure has the following input parameters:

* *replEntinfList*: The [**replicated update**](#gt_2a923099-db0a-4932-af28-4354601e85c4) to be applied.
* *sourcePrefixTable*: The [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) from the server to translate [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) IDs.
* *nc*: The root of the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) that is replicated.
* *attributesAndStamps*: The AttributeAndStamp set that corresponds to the replicated update.

The method returns a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if it encounters an error while updating the object.

1. parentObject: DSNAME
2. newParentObject: DSNAME
3. parentObject := select one o from all-ts-included where
4. (o!objectGUID = replEntinfList.pParentGuid^)
5. if (parentObject = null) then
6. /\* The client will stop processing the reply message. It will
7. \* resend the IDL\_DRSGetNCChanges request with DRS\_GET\_ANC set in
8. \* ulFlags. It is an error for this condition to occur if the
9. \* request already included DRS\_GET\_ANC in ulFlags.
10. \*/
11. return ERROR\_DS\_DRA\_MISSING\_PARENT
12. endif
13. if (not GetObjectNC(parentObject) = nc) then
14. /\* If parentObject exists in an NC replica other than that
15. \* being replicated, the client stops processing the response.
16. \* This condition indicates that parentObject has moved from one
17. \* NC replica to another and that update has not yet been applied
18. \* to the client NC replica containing parentObject.
19. \* This will be rectified when the client replicates the NC
20. \* replica containing parentObject.
21. \*/
22. return ERROR\_DS\_DRA\_OBJ\_NC\_MISMATCH
23. endif
24. /\* Find an appropriate parent object for the object. If the parent
25. \* object is deleted and if the new object is not a deleted object
26. \* then FindBestParentObject will return DSName of "Lost and Found
27. \* container". Otherwise, the parent object will remain the same. \*/
28. newParentObject := FindBestParentObject(parentObject, replEntinfList,
29. sourcePrefixTable, nc, attributesAndStamps)
30. /\* Check whether there is a name conflict (see [MS-ADTS] section
31. \* 3.1.1) and resolve it before adding the object. \*/
32. newObjectDN := ResolveNameConflict(replEntinfList, newParentObject,
33. sourcePrefixTable, attributesAndStamps)
34. /\* Set the new DN in the ENTINF. \*/
35. Copy the value of newObjectDN to replEntinfList.Entinf.pName^.StringName
36. and update the value in replEntinfList.Entinf.pName^.structLen and
37. replEntinfList.Entinf.pName^.NameLen accordingly.
38. return ERROR\_SUCCESS;

##### AddObject

1. procedure AddObject(
2. replEntinfList: REPLENTINFLIST,
3. sourcePrefixTable: PrefixTable,
4. nc: DSName,
5. attributesAndStamps: set of AttributeAndStamp): DWORD

*Informative summary of behavior*: The AddObject procedure performs a [**replicated update**](#gt_2a923099-db0a-4932-af28-4354601e85c4) by adding an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). This procedure has the following input parameters:

* *replEntinfList*: The replicated update to be applied.
* *sourcePrefixTable*: The [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) from the server to translate [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) IDs.
* *nc*: The root of the NC replica that is replicated.
* *attributesAndStamps*: The [AttributeAndStamp](#Section_1cf26e6ee5bf45238c839ade3077a14e) set that corresponds to the replicated update.

The procedure returns a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if it encounters an error while adding the object.

1. newObject: DSName
2. dwResult: DWORD
3. objectClassAttr: ATTRVAL
4. isDeletedAttr: ATTRVAL
5. isDeletedValue: boolean
6. ncNameValue: DSName
7. ncNameAtt: ATTRVAL
8. partitionsContainer: DSName
9. parentObject: DSName
10. /\* Find an appropriate and unused name for the object, updating
11. \* replEntInfList as appropriate \*/
12. dwResult := NameObject(replEntInfList, nc, sourcePrefixTable, attributeAndStamps)
13. if dwResult ≠ ERROR\_SUCCESS then
14. return dwResult
15. endif
16. /\* Check if this is a cross-ref in the partitions container replicating in.\*/
17. objectClassAttr := ENTINF\_GetValue(
18. replEntinfList.Entinf,
19. objectClass,
20. sourcePrefixTable)
21. ncNameAtt := ENTINF\_GetValue(
22. replEntinfList.Entinf,
23. ncName,
24. sourcePrefixTable)
25. ncNameVal := ValueFromATTRVal(ncNameAtt,Syntax(ncNameAtt),sourcePrefixTable)
26. partitionsContainer:= DescendantObject(ConfigNC(), "CN=Partitions,")
27. parentObject := replEntinfList.Entinf.pName^ stripped of the first RDN.
28. if(crossRef in ObjectClassAttr and parentObject = partitionsContainer)
29. dwResult := AddSubRef(ncNameVal)
30. if dwResult ≠ 0 then
31. return dwResult
32. endif
33. isDeletedAttr := ENTINF\_GetValue(
34. replEntinfList.Entinf,
35. isDeleted,
36. sourcePrefixTable)
37. if (isDeletedAttr = null) then
38. isDeletedValue := false
39. else
40. isDeletedValue :=
41. ValueFromATTRVal(isDeletedAttr,Syntax(isDeleted),sourcePrefixTable)
42. endif
43. If(isDeleted Value)
44. DelSubRef(ncNameVal)
45. endif
46. endif
47. dwResult := PerformAddOperation(replEntinfList.Entinf, newObject,
48. sourcePrefixTable, FALSE)
49. /\* Update attribute stamps. \*/
50. if (dwResult = 0) then
51. for each e in attributesAndStamps do
52. SetAttrStamp(newObject, e.attribute, e.stamp)
53. endfor
54. endif
55. return dwResult

##### UpdateObject

1. procedure UpdateObject(
2. replEntinfList: REPLENTINFLIST,
3. sourcePrefixTable: PrefixTable,
4. nc: DSName,
5. attributesAndStamps: set of AttributeAndStamp): DWORD

*Informative summary of behavior*: The UpdateObject procedure performs a [**replicated update**](#gt_2a923099-db0a-4932-af28-4354601e85c4) by applying changes on an existing [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). This procedure has the following input parameters:

* *replEntinfList*: The replicated update to be applied.
* *sourcePrefixTable*: The [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) from the server to translate [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) IDs.
* *nc*: The root of the NC replica that is replicated.
* *attributesAndStamps*: The [AttributeAndStamp](#Section_1cf26e6ee5bf45238c839ade3077a14e) set that corresponds to the replicated update.

The method returns a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if encounters an error while updating the object.

1. updateObject: DSName
2. stampRemote: AttributeStamp
3. stampLocal: AttributeStamp
4. attribute: ATTRTYP
5. nameAttrAndStamp: AttributeAndStamp
6. attrAndStamp: AttributeAndStamp
7. isDeletedAttrAndStamp: AttributeAndStamp
8. dwResult: DWORD
9. updateObject := replEntinfList.Entinf.pName^
10. /\* Determine if attributesAndStamps indicates a rename operation. \*/
11. nameAttrAndStamp := select one e from attributesAndStamps where
12. (e.attribute = name)
13. if (nameAttrAndStamp = null) then
14. stampRemote := null
15. else
16. stampRemote := nameAttrAndStamp.stamp
17. endif
18. stampLocal := AttrStamp(updateObject, name)
19. if (not stampRemote = null) and
20. (AttributeStampCompare(stampRemote, stampLocal) > 0) then
21. /\* This indicates that replEntinfList provides a more recent
22. \* DN for updateObject. It is important to note here that a change
23. \* in the name attribute is interpreted as a potential change in
24. \* the full DN, not just the RDN. \*/
25. /\* The NameObject function will find an appropriate, unused, local
26. \* name for the object and modify the replEntInfList appropriately \*/
27. dwResult := NameObject(replEntInfList, sourcePrefixTable,
28. nc, attributeAndStamps)
29. if dwResult ≠ ERROR\_SUCCESS then
30. return dwResult
31. endif
32. /\* Perform modify operation. \*/
33. /\* Compare local and remote attribute stamps and update object
34. \* attribute only if the changes are more recent than what the
35. \* client has seen. \*/
36. for i := 0 to (replEntinfList.Entinf.AttrBlock.attrCount-1)
37. attribute := LocalAttidFromRemoteAttid(
38. sourcePrefixTable,
39. replEntinfList.Entinf.AttrBlock.pAttr[i].attrTyp);
40. attrAndStamp := select one e from attributeAndStamps where
41. (e.attribute = attribute)
42. stampRemote := attrAndStamp.stamp
43. stampLocal := AttrStamp(updateObject, attribute)
44. if (not stampLocal = null) and
45. (AttributeStampCompare(stampRemote, stampLocal) <= 0) then
46. /\* This indicates the attribute on the object in the client is
47. \* more up to date. Do not apply the replicated update
48. \* corresponding to that attribute.
49. \*/
50. ENTINF\_SetValue(replEntinfList.Entinf, attribute, null,
51. sourcePrefixTable)
52. attributesAndStamps := attributesAndStamps - {attrAndStamp}
53. endif
54. endfor
55. dwResult := PerformModifyOperation(replEntinfList.Entinf,
56. updateObject,
57. sourcePrefixTable
58. )
59. if dwResult ≠ ERROR\_SUCCESS then
60. return dwResult
61. endif
62. /\* Update attribute stamps on the object to those corresponding to
63. \* the replicated updates. \*/
64. for each e in attributesAndStamps do
65. SetAttrStamp(updateObject, e.attribute, e.stamp)
66. endfor
67. if updateObject!isDeleted = true then
68. if(crossRef in updateObject!objectClass)
69. /\* If this is a cross-ref being deleted, then the respective
70. \* sub-ref object, if any, must also be deleted.\*/
71. DelSubRef (updateObject!ncName)
72. endif
73. /\* There might be attribute values left on this object that do not
74. \* conform to the invariants of a tombstone or deleted-object (see
75. \* MS-ADTS section 3.1.1.5.5). Delete the object again to create an
76. \* originating change of any such attribute values that need it.
77. \* This originating change will affect the metadata of updateObject,
78. \* and can explicitly affect metadata just written to the database
79. \* in the above SetAttrStamp procedure. \*/
80. dwResult := RemoveObj(updateObject,false)
81. else
82. isDeletedAttrAndStamp := select one e from attributesAndStamps where
83. (e.attribute = isDeleted)
84. if(isDeletedAttrAndStamp != null and crossRef in updateObject!objectClass)
85. /\* If this is a cross-ref being undeleted, then we must also undelete
86. \* the respective sub-ref object. \*/
87. AddSubRef (updateObject!ncName)
88. endif
89. endif
90. if updateObject!isRecycled = true and IsRecycleBinEnabled() then
91. /\* There might be attribute values left on this object that do
92. \* not conform to the invariants of a recycled-object (see MS-ADTS
93. \* section 3.1.1.5.5). Recycle the object again to create an originating
94. \* change of any such attribute values that need it. This
95. \* originating change will affect the metadata of updateObject, and
96. \* can explicitly affect metadata just written to the database in
97. \* the above SetAttrStamp procedure. \*/
98. dwResult := RecycleObj(updateObject)
99. endif
100. return dwResult

##### FindBestParentObject

1. procedure FindBestParentObject(
2. parentObject: DSName,
3. replEntinfList: REPLENTINFLIST,
4. sourcePrefixTable: PrefixTable,
5. nc: DSName,
6. var attributesAndStamps: set of AttributeAndStamp): DSName

*Informative summary of behavior*: Given a desired [**parent object**](#gt_0d41951a-62f0-4fbd-bb23-22f645ae3bf5), the FindBestParentObject procedure validates whether the desired parent object is deleted. If the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that is being [**updated**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) is not a deleted object and the desired parent object is deleted, this procedure returns the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the [**Lost and Found container**](#gt_2761433f-c664-4b7e-af5e-e16ab6d86c5c).

Following are the input parameters for this procedure:

* *parentObject*: The DSName of the desired parent object.
* *replEntinfList*: The [**replicated update**](#gt_2a923099-db0a-4932-af28-4354601e85c4) that is to be applied.
* *sourcePrefixTable*: The [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) from the server.
* *nc*: The DSName of the root of the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).
* *attributesAndStamps*: The [AttributeAndStamp](#Section_1cf26e6ee5bf45238c839ade3077a14e) set that corresponds to the replicated update (can be modified by this procedure).

1. isDeletedAttr: ATTRVAL
2. isDeletedValue: boolean
3. attrAndStamp: AttributeAndStamp
4. isDeletedAttr := ENTINF\_GetValue(
5. replEntinfList.Entinf,
6. isDeleted,
7. sourcePrefixTable)
8. if (isDeletedAttr = null) then
9. isDeletedValue := false
10. else
11. isDeletedValue := ValueFromATTRVal(
12. isDeletedAttr, Syntax(isDeleted), sourcePrefixTable)
13. endif
15. if isDeletedValue = false and parentObject!isDeleted = true then
16. /\* This indicates that an object was moved/created under
17. \* parentObject in one NC replica while parentObject was deleted
18. \* in another NC replica. In this case move/add an object under
19. \* the "lost and found" container.
20. \*/
22. /\* Remove attribute stamp for name so that the update is seen
23. \* as an originating update. \*/
24. attrAndStamp := select one from attributesAndStamps where
25. (e.attribute = name)
26. attributesAndStamps := attributesAndStamps - {attrAndStamp}
27. return GetWellKnownObject(nc, GUID\_LOSTANDFOUND\_CONTAINER\_W)
28. endif
29. return parentObject

##### ResolveNameConflict

1. procedure ResolveNameConflict(
2. replEntinfList: REPLENTINFLIST,
3. parentObject: DSName,
4. var attributesAndStamps: set of AttributeAndStamp): DN

*Informative summary of behavior*: The ResolveNameConflict procedure checks whether there is a name conflict (see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1) while applying a [**replicated update**](#gt_2a923099-db0a-4932-af28-4354601e85c4). If there is a name conflict, the procedure changes the desired [DN](#Section_837c7001335148dea177c165a584816c) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for which the replicated update is applied, or changes the DN of the existing object so that there is no name conflict. Following are the input parameters for this procedure:

*replEntinfList*: The [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) to be applied.

*parentObject*: The [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the [**parent object**](#gt_0d41951a-62f0-4fbd-bb23-22f645ae3bf5).

*attributesAndStamps*: The [AttributeAndStamp](#Section_1cf26e6ee5bf45238c839ade3077a14e) set that corresponds to the replicated update (can be modified by this procedure).

1. objectRDN: RDN
2. objectDN: DN
3. rdnValue: unicodestring
4. duplicateObject: DSName
5. nameAttrStamp: AttributeAndStamp
6. guidUpdateObj: GUID
7. stampExistingObj: AttributeStamp
8. stampUpdateObj: AttributeStamp
9. objectRDN := leftmost RDN of replEntinfList.Entinf.pName^.StringName
10. rdnValue := AttributeValue portion of objectRDN (see [RFC2253])
11. objectDN := objectRDN followed by RDNs of
12. parentObject!distinguishedName
13. duplicateObject := select one d from children parentObject where
14. (d!name = rdnValue)
15. if (not duplicateObject = null) and
16. (not duplicateObject!objectGUID =
17. replEntinfList.Entinf.pName^.Guid^) then
18. /\* There already exists a child object (duplicateObject) of
19. \* parentObject whose name attribute value will be same as the name
20. \* attribute value of the object being renamed/added. \*/
21. guidUpdateObj := replEntinfList.Entinf.pName^.Guid^
22. nameAttrStamp := select v from attributesAndStamps where
23. (v.attribute = name)
24. stampUpdateObj := nameAttrStamp.stamp
25. stampExistingObj := AttrStamp(duplicateObject, name)
26. if (stampExistingObj.timeChanged > stampUpdateObject.timeChanged)
27. or ((stampExistingObj.timeChanged =
28. stampUpdateObject.timeChanged)
29. and (existingObject!objectGUID > guidUpdateObject)) then
30. /\* Rename the replicated object. \*/
31. newDN = MakeConflictDN(objectDN, guidUpdateObj)
32. /\* Remove existing attribute stamp for name and add in a new one
33. \* so that the update is seen as an originating update. \*/
34. attributesAndStamps := attributesAndStamps - {nameAttrStamp}
35. nameAttrStamp := originating update stamp
36. /\* See MS-ADTS section 3.1.1.1.9 \*/
37. attributeAndStamps := attributeAndStamps + {nameAttrStamp}
38. return newDN
39. else
40. /\* Rename the existing object. \*/
41. newDN = MakeConflictDN(
42. existingObject!distinguishedName,
43. existingObject!objectGUID)
44. newRDN = The left most RDN of newDN
45. PerformModifyDNOperation(existingObject!distinguishedName,
46. null, newRDN)
47. return objectDN
48. endif
49. else
50. return objectDN /\* No conflict case \*/
51. endif

##### MakeConflictDN

1. procedure MakeConflictDN(oldDN: DN, guid: GUID): DN

The MakeConflictDN procedure is used during name conflict resolution. For more details, see section [4.1.10.6.12](#Section_684fdad2cda644259dfd14cd23ec540c).

A conflict name for [DN](#Section_837c7001335148dea177c165a584816c) *oldDN* and [GUID](#Section_5e740f50e6a048c9bca800072e85d963) *guid* is the DN *newDN*, such that *newDN* is the same as *oldDN* with the exception of the AttributeValue portion (as specified in [[RFC2253]](https://go.microsoft.com/fwlink/?LinkId=90327)) of the first [RDN](#Section_042a097c68374e16b04fdd81e65f6613). This portion is the concatenation of:

* The AttributeValue portion of the first RDN of *oldDN*.
* The [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) character 0x000A.
* The Unicode string "CNF:".
* The dashed string representation of guid.

For example, given *oldDN* = "CN=Engineering,DC=Fabrikam,DC=com" and *guid* = a746b716-0ac0-11d2-b376-0000f87a46c8, *newDN* is "CN=Engineering#CNF:a746b716-0ac0-11d2-b376-0000f87a46c8,DC=Fabrikam,DC=com", where the # represents the Unicode character 0x000A.

The procedure returns *newDN*.

##### ProcessLinkValue

1. procedure ProcessLinkValue(
2. replValinf: REPLVALINF\_NATIVE,
3. nc: DSName,
4. sourcePrefixTable: PrefixTable,
5. ulFlags: ULONG,
6. ulMoreFlags: ULONG): DWORD

*Informative summary of behavior*: The ProcessLinkValue procedure applies the [**replicated update**](#gt_2a923099-db0a-4932-af28-4354601e85c4) of a [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238). Following are the input parameters for this procedure.

* *replValinf*: The link value replicated update.
* *nc*: The [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the root of the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) where the replicated update is applied.
* *sourcePrefixTable*: The [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) from the server.
* *ulFlags*: A [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) bit field.
* *ulMoreFlags*: A [DRS\_MORE\_GETCHGREQ\_OPTIONS](#Section_d16a3483d4e34f7eb4daed8cded8d970) bit field.

1. updateObject: DSName
2. targetObject: DSName
3. isDeleted: boolean
4. attribute: ATTRTYP
5. attributeValue: attribute value
6. attributeValues: set of attribute value
7. newAttributeValue: attribute value
8. localValueStamp: LinkValueStamp
9. remoteValueStamp: LinkValueStamp
10. updateObject := replValinf.pObject^
11. if (not ObjExists(updateObject)) then
12. /\* The client will stop processing the reply message. It will
13. \* resend the IDL\_DRSGetNCChanges request with DRS\_GET\_ANC set
14. \* in ulFlags. It is an error for this condition to occur if the
15. \* request already included DRS\_GET\_ANC in ulFlags. \*/
16. return ERROR\_DS\_DRA\_MISSING\_PARENT
17. endif
18. if (IsRecycleBinEnabled()) then
19. isRecycled := updateObject!isRecycled
20. if (isRecycled = true) then
21. if (DRS\_GET\_ANC in ulFlags) then
22. /\* Local object is recycled, and it is up-to-date.
23. Replicated update is not applied on a recycled object \*/
24. return 0
25. else
26. /\* Local object is recycled, but it might not be up-to-date. \*/
27. return ERROR\_DS\_DRA\_MISSING\_PARENT
28. endif
29. endif
30. else
31. isDeleted := updateObject!isDeleted
32. if (isDeleted = true) then
33. if (DRS\_GET\_ANC in ulFlags) then
34. /\* Local object is deleted, and it is up-to-date.
35. Replicated update is not applied on a deleted object.\*/
36. return 0
37. else
38. /\* Local object is deleted, but it might not be up-to-date. \*/
39. return ERROR\_DS\_DRA\_MISSING\_PARENT
40. endif
41. endif
42. endif
43. attribute := replValinf.attrTyp
44. attributeValues := GetAttrVals(updateObject, attribute, true)
45. attributeValue := select one k from attributeValues where
46. (k = ValueFromATTRVAL(
47. sourcePrefixTable, Syntax(attribute), replValInf.pAval))
48. if (attributeValue = null) then
49. localValueStamp := null
50. else
51. /\* If attributeValue was last updated when the forest functional
52. \* level was DS\_BEHAVIOR\_WIN2000, no LinkValueStamp is
53. \* associated with attributeValue. In that case the procedure
54. \* LinkStamp() returns null.
55. \*/
56. localValueStamp :=
57. LinkStamp(updateObject, attribute, attributeValue)
58. endif
59. remoteValueStamp := AbstractLinkValStampFromConcreteLinkValStamp(
60. replValinf.MetaData)
61. if (localValueStamp = null) or
62. (LinkValueStampCompare(localValueStamp, remoteValueStamp) < 0)
63. then
64. /\* The replicated update is more up to date. Apply that change and
65. \* modify the stamp. \*/
66. newAttributeValue = ValueFromATTRVAL(
67. sourcePrefixTable, Syntax(attribute), replValInf.pAval)
68. targetObject := GetDSNameFromAttrVal( replValinf.attrTyp, replValInf.pAval)
69. if (targetObject = null)
70. return ERROR\_DS\_INVALID\_ATTRIBUTE\_SYNTAX
71. if ((IsRecycleBinEnabled() and targetObject!isRecycled) or
72. (not IsRecycleBinEnabled() and targetObject!isDeleted)) then
73. if (DRS\_GET\_TGT in ulMoreFlags) then
74. /\* nothing to do \*/
75. return 0
76. else
77. return ERROR\_DS\_DRA\_RECYCLED\_TARGET
78. endif
79. if (not attributeValue = null) then
80. /\* Remove the old attribute value. \*/
81. RemoveAttrVal(updateObject, attribute, attributeValue)
82. endif
83. SetAttrVal(updateObject, attribute, newAttributeValue)
84. /\* If the abstract variable timeDeleted associated with the
85. \* attribute value has a non-zero value, it indicates that the
86. \* value has been deleted from the NC replica. \*/
87. if (replValInf.fIsPresent = false) then
88. remoteValueStamp.timeDeleted := current time on the client
89. else
90. remoteValueStamp.timeDeleted := 0
91. endif
92. SetLinkStamp(updateObject, attribute, newAttributeValue,
93. remoteValueStamp)
94. endif
95. return 0

##### UpdateRepsFrom

1. procedure UpdateRepsFrom(
2. rf: RepsFrom,
3. msgReplyNative: DRS\_MSG\_GETCHGREPLY\_NATIVE,
4. dsaServer: DSName,
5. ulResult: DWORD)

*Informative summary of behavior*: Using the UpdateRepsFrom procedure, the client [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the [repsFrom](#Section_3ef27d3cb9c944048e53ebf3a64a9a10) abstract variable after it applies the response message received from the server. Following are the input parameters for this procedure.

* *rf:* The RepsFrom for the server.
* *msgReplyNative*: The [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) response from the server.
* *dsaServer*: The [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of the server.
* *ulResult*: A [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) that indicates whether or not the [**replicated updates**](#gt_2a923099-db0a-4932-af28-4354601e85c4) in the response message are applied successfully.

1. rfOld: RepsFrom
2. currentTime: DSTIME
3. nc: DSName
4. nc := msgReplyNative.pNC^
5. rfOld := select one v from nc!repsFrom where
6. (v.uuidDsa = dsaServer!objectGUID)
7. if rfOld ≠ null then
8. nc!repsFrom := nc!repsFrom - {rfOld} /\* remove old entry \*/
9. endif
10. currentTime := current time on the client
11. rf.timeLastAttempt := currentTime
12. if (ulResult = 0) then
13. rf.consecutiveFailures := 0
14. rf.timeLastSuccess := currentTime
15. rf.resultLastAttempt := 0
16. rf.uuidInvocId := msgReplyNative.uuidInvocIdSrc
17. rf.usnVec := msgReplyNative.usnvecTo
18. rf.resultLastAttempt := 0
19. else
20. rf.consecutiveFailures := rf.consecutiveFailures + 1
21. rf.resultLastAttempt := ulResult
22. endif
23. nc!repsFrom := nc!repsFrom + {rfNew}

##### UpdateUTDandPAS

1. procedure UpdateUTDandPAS(
2. msgReplyNative: DRS\_MSG\_GETCHGREPLY\_NATIVE,
3. partialAttrSetEx: PARTIAL\_ATTR\_VECTOR\_V1\_EXT,
4. nc: DSName)

*Informative summary of behavior*: If the client has applied all [**replicated updates**](#gt_2a923099-db0a-4932-af28-4354601e85c4) in the response message of [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) from the server, and if the [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16) is complete, then the client [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the replUpToDateVector and partialAttributeSet abstract [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f), as specified in the UpdateUTDandPAS procedure. This procedure has the following input parameters.

* *msgReplyNative*: The IDL\_DRSGetNCChanges response from the server.
* *partialAttrSetEx*: The [PARTIAL\_ATTR\_VECTOR\_V1\_EXT](#Section_1d5c1b34daa44761a8b5d3c0146a0e30) structure that contains attributes to be added to the partialAttributeSet abstract variable.
* *nc*: The [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the root of the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) where the replicated update is applied.

1. partialAttrSetAdd: sequence of ATTRTYP
2. remoteCursor: UPTODATE\_CURSOR\_V2
3. localCursor: ReplUpToDateVector
4. newCursor: ReplUpToDateVector
5. nc: DSName
6. i: DWORD
7. nc := msgReplyNative.pNC^
8. /\* Update partialAttributeSet abstract attribute. \*/
9. if (not partialAttrSetEx.cAttrs = 0) then
10. partialAttrSetAdd = AbstractPASFromConcretePAS(partialAttrSetEx)
11. nc!partialAttributeSet :=
12. nc!partialAttributeSet + partialAttrSetAdd
13. endif
14. /\* Merge replUpToDateVector abstract attribute \*/
15. for i := 0 to msgReplyNative.pUpToDateVecSrc^.cNumCursors - 1
16. remoteCursor := msgReplyNative.pUpToDateVecSrc^.rgCursors[i]
17. localCursor := select one v from nc!replUpToDateVector where
18. (v.uuidDsa = remoteCursor.uuidDsa)
19. if (localCursor = null) then
20. /\* An entry for the server does not exist; add it. \*/
21. newCursor.uuidDsa := remoteCursor.uuidDsa
22. newCursor.usnHighPropUpdate := remoteCursor.usnHighPropUpdate
23. newCursor.timeLastSyncSuccess := remoteCursor.timeLastSyncSuccess
24. nc!replUpToDateVector := nc!replUpToDateVector + {newCursor}
25. else
26. /\* Update existing entry for the server. \*/
27. if (localCursor.usnHighPropUpdate <
28. remoteCursor.usnHighPropUpdate) then
29. newCursor.usnHighPropUpdate := remoteCursor.usnHighPropUpdate
30. newCursor.timeLastSyncSuccess :=
31. remoteCursor.timeLastSyncSuccess
32. newCursor.uuidDsa := remoteCursor.uuidDsa
33. nc!replUpToDateVector :=
34. nc!replUpToDateVector - {localCursor} + {newCursor}
35. endif
36. endif
37. endfor
38. return

##### DecryptValuesIfNecessary

1. procedure DecryptValuesIfNecessary(
2. hDrs: DRS\_HANDLE,
3. prefixTable: PrefixTable,
4. var attrBlock: ATTRBLOCK): DWORD

*Informative summary of behavior*: The values of several [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) are encrypted by the server and conversely have to be decrypted by the client before processing [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493). The client decrypts the encrypted data by using MD5 [**digest**](#gt_1c222b9e-7176-4840-9d19-e65895b9fc62) (as specified in [[RFC1321]](https://go.microsoft.com/fwlink/?LinkId=90275)), a [**CRC32**](#gt_9cb45a36-92bb-4c14-b2fd-2ad7e2979bfd) [**checksum**](#gt_fa444149-ef93-4512-a278-2e756295630c) (as specified in [[ISO/IEC 13239]](https://go.microsoft.com/fwlink/?LinkId=98149)), and RC4 stream cipher (as specified in [[RC4]](https://go.microsoft.com/fwlink/?LinkId=93759)). The DecryptValuesIfNecessary procedure specifies the process of attribute value decryption.

Following are the input parameters for this method.

* *hDrs*: The [DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) derived by sending [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) to the server.
* *prefixTable*: The [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) used to translate attribute IDs.
* *attrBlock*: The [ATTRBLOCK](#Section_f81324b8640041b5bc255117589c602a) structure that is derived from the response of the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) message. If *attrBlock* has attribute values that need to be decrypted, then the values are decrypted in place. That is, at the end of the procedure call, the **pVal** field in the [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) structure refers to the decrypted attribute value.

The procedure returns a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) on failure. Otherwise, it returns 0.

1. localAttid: ATTRTYP
2. attr: ATTR
3. pPayload: ADDRESS OF ENCRYPTED\_PAYLOAD
4. salt: sequence of BYTE
5. sessionKey: sequence of BYTE
6. i: integer
7. j: integer
8. crcComputed: ULONG
9. crcReceived: ULONG
10. md5Context: MD5\_CTX
11. /\* Get session key associated with the RPC connection. \*/
12. sessionKey := session key associated with security context of hDrs,
13. as specified by [MS-RPCE] section 3.3.1.5.2, "Building and Using a
14. Security Context", and [MS-KILE] section 3.1.1.2, "Cryptographic
15. Material"
16. for j := 0 to (attrBlock.attrCount - 1)
17. attr := attrBlock.pAttr[j]
18. localAttid = LocalAttidFromRemoteAttid(prefixTable, attr.attrTyp)
19. if IsSecretAttribute(localAttid) then
20. /\* Decrypt all values of this attribute. \*/
21. for i := 0 to (attr.AttrVal.valCount - 1)
22. pPayload := attr.AttrVal.pAVal[i].pVal
23. salt := pPayload^.Salt
24. /\* Compute encryption key. \*/
25. MD5Init(md5Context)
26. MD5Update(md5context, sessionKey, sessionKey.length)
27. MD5Update(md5context, salt, 16)
28. MD5Final(md5Context)
29. Decrypt (attr.AttrVal.pAVal[i].valLen - 16) bytes starting at
30. the address of pPayload^.Checksum using the RC4 stream cipher
31. algorithm [RC4] with encryption key md5Context.digest. At the
32. end of this operation pPayload^.EncryptedData field contains
33. decrypted attribute value.
34. /\* Calculate checksum of the clear value. \*/
35. crcComputed :=
36. CRC32 [ISO/IEC 13239] of the
37. (attr.AttrVal.pAVal[i].valLen - 20)
38. bytes starting at pPayload^.EncryptedData
39. crcReceived := pPayload^.Checksum
40. if (not crcComputed = crcReceived) then
41. /\* Checksums don't match. Stop processing the reply message.
42. \*/
43. return SEC\_E\_ALGORITHM\_MISMATCH
44. endif
45. /\* Modify ATTRVAL structure to have decrypted data. \*/
46. attr.AttrVal.pAVal[i].valLen :=
47. attr.AttrVal.pAVal[i].valLen - 20
48. attr.AttrVal.pAVal[i].pVal := ADR(pPayload^.EncryptedData)
49. endfor
50. endif
51. endfor
52. return 0

##### DecompressReplyMessage

1. procedure DecompressReplyMessage(
2. msgOut: DRS\_MSG\_GETCHREPLY,
3. dwOutVersion: DWORD): DRS\_MSG\_GETCHGREPLY\_NATIVE

*Informative summary of behavior*: Compression subdivides a data stream into sequences of bytes called [**compression chunks**](#gt_406bb66c-9e36-4bfe-9901-e852b24d2e0d). The DecompressReplyMessage procedure decompresses the data stream.

1. pInBuffer: sequence of BYTE
2. pOutBuffer: sequence of BYTE
3. cbInBufferCompress: DWORD
4. cbInBufferDeCompress: DWORD
5. if (dwOutVersion = 2) or
6. (dwOutVersion = 7) then
7. /\* decompress data that is compressed.
8. \*/
9. if (dwOutVersion = 2) then
10. pInBuffer := msgOut.V2.CompressedV1.pbCompressedData
11. cbInBufferCompress := msgOut.CompressedV1.cbCompressedSize
12. cbInBufferDeCompress := msgOut.CompressedV1.cbUncompressedSize
13. compressionAlg := DRS\_COMP\_ALG\_MSZIP
14. else if (dwOutVersion = 7) then
15. pInBuffer := msgOut.V7.CompressedAny.pbCompressedData
16. cbInBufferCompress := msgOut.V7.CompressedAny.cbCompressedSize
17. cbInBufferDecompress :=
18. msgOut.V7.CompressedAny.cbUncompressedSize
19. compressionAlg := msgOut.V7.CompressionAlg
20. endif
21. DecompressMessage(pInBuffer, cbInBufferCompress, cbOutBufferCompress,
22. compressionAlg, pOutputBuffer)
23. /\* pOutputBuffer now has the uncompressed data that was derived by
24. \* serializing a DRS\_GETCHGREPLY structure at the server.
25. \* Convert the serialized data back to DRS\_GETCHGREPLY structure.\*/
26. if dwOutVersion = 2 then
27. dwOutVersion := 1
28. else
29. dwOutVersion := msgOut.V7.dwCompressedVersion
30. endif
31. msgOut := Unpickling of data in pOutBuffer of length cbOutBuffer,
32. as specified by [C311] Part 2, "IDL/NDR Pickles", and
33. [MS-RPCE] sections 2.2.6 and 2.2.7, "Type Serialization
34. Version 1" and "Type Serialization Version 2"
35. endif
36. return GetNCChangesNativeReply(msgOut, dwOutVersion)

##### DecompressMessage

1. procedure DecompressMessage(
2. pInBuffer: sequence of BYTE,
3. cbInBufferCompress: DWORD,
4. cbInBufferDecompress: DWORD,
5. DRS\_COMP\_ALG\_TYPE: compressionAlg
6. ref pOutputBuffer: sequence of BYTE)

*Informative summary of behavior*: Compression subdivides a data stream into sequences of bytes called [**compression chunks**](#gt_406bb66c-9e36-4bfe-9901-e852b24d2e0d). The DecompressMessage procedure decompresses the data stream.

The following table identifies the size of the compression chunk for each algorithm type.

| Algorithm | Chunk size |
| --- | --- |
| COMP\_ALG\_NONE | Not applicable |
| COMP\_ALG\_MSZIP | 32768 |
| COMP\_ALG\_W2K3 | 65536 |

Each compression chunk in the compressed byte sequence is represented by means of a [COMPRESSED\_DATA](#Section_e4380043164748699d683c0fefa5edd7) structure.

1. pInBlock: ADDRESS OF COMPRESSED\_DATA
2. cbInputProcessed: DWORD
3. cbDecompressedData: DWORD
4. if (cbInBufferCompress = cbInBufferDecompress) then
5. /\* No decompression required here. \*/
6. pOutBuffer := pInBuffer
7. cbOutBuffer := cbInBufferDeCompress
8. else
9. cbInputProcessed := 0
10. while (cbInputProcessed ≤ cbInBufferCompress)
11. pInBlock := ADR(pInputBuffer[cbInputProcessed])
12. if (pInBlock^.cbDecompressedSize =
13. pInBlock^.cbCompressedSize) then
14. pDecompressedData := pInBlock^.data
15. cbDecompressedData := pInBlock^.cbDecompressedSize
16. else
17. if (compressionAlg = DRS\_COMP\_ALG\_MSZIP) then
18. pDecompressedData :=
19. Decompress pInBlock^.data in accordance
20. with [RFC1951].
21. else
22. pDecompressedData := new sequence of BYTE of length
23. pInBlock^.cbDecompressedSize
24. CompressOrDecompressWin2k3(pInBlock^.data,
25. pInBlock^.cbDecompressedSize,
26. pDecompressedData, FALSE)
27. endif
28. cbDecompressedData := pInBlock^.cbDecompressedSize
29. endif
30. pOutputBuffer := Append sequence of BYTE pDeCompressedData of
31. size cbDecompressedData to sequence of BYTE
32. pOutputBuffer
33. cbOutputBuffer :=
34. cbOutputBuffer + pInBlock^.cbDecompressedSize
35. cbInputProcessed := cbInputProcessed +
36. pInBlock^.cbCompressedSize
37. Round up value in cbInputProcessed such that
38. ADR(pInBlock[cbInputProcessed]) align on double word
39. boundary.
40. endwhile
41. endif

#### Examples of the IDL\_DRSGetNCChanges Method - Add User

##### Initial State

User "Kim Akers" is created on DC1 with the sAMAccountName "KimAkers"

ldap\_add\_s("CN=Kim Akers,CN=Users,DC=contoso,DC=com", [*sAMAccountName*])

Added {CN=Kim Akers,CN=Users,DC=contoso,DC=com }.

Querying the nTDSDSA [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for the root domain NC DC=CONTOSO, DC=COM for DC1:

* ldap\_search\_s("CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com", *baseObject*, "(objectClass=\*)", [*objectClass, cn ... objectGUID*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=NTDS Settings,CN=DC1,CN=Servers, CN=Default-First-Site-Name,CN=Sites, CN=Configuration,DC=contoso,DC=com
  + 3> objectClass: top; applicationSettings; nTDSDSA;
  + 1> cn: NTDS Settings;
  + 1> distinguishedName: CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> objectGUID: c20bc312-4d35-4cc0-9903-b1073368af4a;

Querying the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) "CN=Kim Akers, CN=Users, DC=CONTOSO, DC=COM" on DC1:

* ldap\_search\_s("CN=Kim Akers,CN=Users,DC=contoso,DC=com", *baseObject*, "(objectClass=\*)", [*objectClass, cn ... objectCategory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=Kim Akers,CN=Users,DC=contoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> cn: Kim Akers;
  + 1> sn: Dow;
  + 1> givenName: Kim;
  + 1> distinguishedName: CN=Kim Akers,CN=Users,DC=contoso,DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/17/2006 13:50:32 Pacific Standard Pacific Daylight Time;
  + 1> whenChanged: 07/17/2006 13:50:33 Pacific Standard Pacific Daylight Time;
  + 1> displayName: Kim Akers;
  + 1> uSNCreated: 29345;
  + 1> uSNChanged: 29350;
  + 1> name: Kim Akers;
  + 1> objectGUID: 39ab8618-d3fd-410c-b627-64b65104384d;
  + 1> userAccountControl: 0x200 = ( UF\_NORMAL\_ACCOUNT );
  + 1> badPwdCount: 0;
  + 1> codePage: 0;
  + 1> countryCode: 0;
  + 1> badPasswordTime: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogoff: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogon: 01/01/1601 00:00:00 UNC ;
  + 1> pwdLastSet: 07/17/2006 13:50:33 Pacific Standard Time Pacific Daylight Time;
  + 1> primaryGroupID: 513;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1129;
  + 1> accountExpires: 09/14/30828 02:48:05 UNC ;
  + 1> logonCount: 0;
  + 1> sAMAccountName: KimAkers;
  + 1> sAMAccountType: SAM\_NORMAL\_USER\_ACCOUNT;
  + 1> userPrincipalName: KimAkers@contoso.com;
  + 1> objectCategory: CN=Person,CN=Schema,CN=Configuration,DC=contoso,DC=com;

Querying the repsFrom [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root object for [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) DC=CONTOSO, DC=COM on DC2:

* ldap\_search\_s("DC=contoso,DC=com", *baseObject*, "(objectclass=\*)", )
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: DC=contoso,DC=com
  + 1> repsFrom: dwVersion = 1, V1.cb: 276, V1.cConsecutiveFailures: 4
    - V1.timeLastSuccess: 12797642625 V1.timeLastAttempt: 12797643058

V1.ulResultLastAttempt: 0x2108 V1.cbOtherDraOffset: 216

V1.cbOtherDra: 60 V1.ulReplicaFlags: 0x70

* + - V1.rtSchedule: <ldp:skipped> V1.usnvec.usnHighObjUpdate: 29322

V1.usnvec.usnHighPropUpdate: 29322

* + - V1.uuidDsaObj: c20bc312-4d35-4cc0-9903-b1073368af4a

V1.uuidInvocId: c20bc312-4d35-4cc0-9903-b1073368af4a

V1.uuidTransportObj: 00000000-0000-0000-0000-000000000000

* + - V1.mtx\_address: c20bc312-4d35-4cc0-9903-b1073368af4a.\_msdcs.contoso.com
    - V1.cbPASDataOffset: 0

Where V1 represents the [REPS\_FROM](#Section_f8e930ead84745858d58993e05f55e45) structure. V1.mtx\_address represents the [MTX\_ADDR](#Section_107b7c0e0f0d4fe2823214ec3b78f40d) structure stored in the data field of the REPS\_FROM structure.

Querying the user object "CN=Kim Akers, CN=Users, DC=CONTOSO, DC=COM" on DC2 returns no entries because the object is not present on DC2.

* ldap\_search\_s("CN=Kim Akers,CN=Users,DC=contoso,DC=com", *singleLevel*, "(objectclass=\*)", *null*)
* Error: Search: No Such Object.
* Matched DNs: CN=Users,DC=contoso,DC=com
* Getting 0 entries:

##### Client Request

DC2 invokes the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method against DC1, with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 10
* *pmsgIn* = [DRS\_MSG\_GETCHGREQ\_V10](#Section_92b1b77d205846e09e8c6664b96a0cf9)
  + Destination DSA objGuid: \_GUID {6aad8f5a-07cc-403a-9696-9102fe1c320b}
  + Source DSA Invocation ID: \_GUID {c20bc312-4d35-4cc0-9903-b1073368af4a}
  + usnvecFrom: [USN\_VECTOR](#Section_595d11b86ca74a61bd563e6a2b99b76b)
    - usnHighObjUpdate : 29322
    - usnHighPropUpdate : 29322
  + pUpToDateVecDest : [UPTODATE\_VECTOR\_V1\_EXT](#Section_462b424ab50a4c4aa81f48d0f4cf40fe)
    - DSA Invoc ID: 9876730c-5844-4c94-b0bd-28458be39333, USN: 27359
    - DSA Invoc ID: c20bc312-4d35-4cc0-9903-b1073368af4a, USN: 29335
  + ulFlags:
    - DRS\_ASYNC\_OP
    - DRS\_WRIT\_REP
    - DRS\_INIT\_SYNC
    - DRS\_PER\_SYNC
  + Max objects to return: 535
  + Max bytes to return: 5357731
  + Extended operation: none
  + Fsmo Info: 0
  + PrefixTableDest : [SCHEMA\_PREFIX\_TABLE](#Section_9b371267e8b84c69997902dae02e5e38)
  + ulMoreFlags: 0

##### Server Response

Return code of 0 with the following values:

* *pdwOutVersion*= DRS\_MSG\_GETCHGREPLY\_NATIVE\_VERSION\_NUMBER
* *pmsgOut* = [DRS\_MSG\_GETCHGREPLY\_NATIVE](#Section_8079e22efbc04675979cb95cee7f29a5)
  + uuidDsaObjSrc: \_GUID {c20bc312-4d35-4cc0-9903-b1073368af4a}
  + uuidInvocIdSrc: \_GUID {c20bc312-4d35-4cc0-9903-b1073368af4a}
  + pNC: [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) DC=CONTOSO,DC=COM
  + usnvecFrom : [USN\_VECTOR](#Section_595d11b86ca74a61bd563e6a2b99b76b)
    - usnHighObjUpdate : 29322
    - usnHighPropUpdate : 29322
  + usnvecTo: USN\_VECTOR
    - usnHighObjUpdate : 29379
    - usnHighPropUpdate : 29379
  + pUpToDateVecSrc : [UPTODATE\_VECTOR\_V2\_EXT](#Section_cebd1ccb891b4268b0564b714cdf981e)
    - DSA Invoc ID: c20bc312-4d35-4cc0-9903-b1073368af4a,
    - usnHighPropUpdate : 29379, timeLastSyncSuccess : 12797643933
  + PrefixTableSrc : [SCHEMA\_PREFIX\_TABLE](#Section_9b371267e8b84c69997902dae02e5e38)
  + pObjects: [REPLENTINFLIST](#Section_c38b0412cf004b0cb4f44662a4484a00)
    - objectClass: top; person; organizationalPerson; user;
    - sn: Akers;
    - givenName: Kim;
    - instanceType: 0x4 = ( IT\_WRITE );
    - whenCreated: 07/17/2006 13:50:32 Pacific Standard Daylight Time;
    - whenChanged: 07/17/2006 14:05:21 Pacific Standard Daylight Time;
    - displayName: Kim Akers;
    - nTSecurityDescriptor: *binary data*
    - objectGUID: 39ab8618-d3fd-410c-b627-64b65104384d;
    - codePage: 0;
    - countryCode: 0;
    - dBCSPwd: *binary data*
    - logonHours: 0
    - unicodePwd: *binary data*
    - ntPwdHistory: *binary data*
    - pwdLastSet: 07/17/2006 13:50:33 Pacific Standard Daylight Time;
    - sAMAccountName: KimAkers;
    - sAMAccountType: SAM\_NORMAL\_USER\_ACCOUNT;
    - userPrincipalName: KimAkers@contoso.com;
    - objectCategory: CN=Person, CN=Schema, CN=Configuration, DC=contoso, DC=com;
  + rgValues: (null)

##### Final State

Querying the repsFrom [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) DC=CONTOSO, DC=COM on DC2:

* ldap\_search\_s("DC=contoso,DC=com", *baseObject*, "(objectclass=\*)", *repsFrom*)
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: DC=contoso,DC=com
  + 1> repsFrom: dwVersion = 1, V1.cb: 276, V1.cConsecutiveFailures: 0
    - V1.timeLastSuccess: 12797643933 V1.timeLastAttempt: 12797643933 V1.ulResultLastAttempt: 0x0 V1.cbOtherDraOffset: 216
    - V1.cbOtherDra: 60 V1.ulReplicaFlags: 0x70
    - V1.rtSchedule: <ldp:skipped> V1.usnvec.usnHighObjUpdate: 29379 V1.usnvec.usnHighPropUpdate: 29379
    - V1.uuidDsaObj: c20bc312-4d35-4cc0-9903-b1073368af4a
    - V1.uuidInvocId: c20bc312-4d35-4cc0-9903-b1073368af4a V1.uuidTransportObj: 00000000-0000-0000-0000-000000000000 V1.mtx\_address: c20bc312-4d35-4cc0-9903-b1073368af4a.\_msdcs.contoso.com
    - V1.cbPASDataOffset: 0 V1.PasData: version = -1, size = -1, flag = -1

Querying the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) "CN=Kim Akers, CN=Users, DC=CONTOSO,DC=COM" on DC2, which is now present:

* ldap\_search\_s("CN=Kim Akers,CN=Users,DC=contoso,DC=com", *baseObject*, "(objectclass=\*)", [*objectClass, cn ... objectCategory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=Kim Akers,CN=Users,DC=contoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> cn: Kim Akers;
  + 1> sn: Akers;
  + 1> givenName: Kim;
  + 1> distinguishedName: CN=Kim Akers,CN=Users,DC=contoso,DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/17/2006 13:50:32 Pacific Standard Daylight Time;
  + 1> whenChanged: 07/17/2006 14:05:21 Pacific Standard Daylight Time;
  + 1> displayName: Kim Akers;
  + 1> uSNCreated: 38197;
  + 1> uSNChanged: 38197;
  + 1> name: Kim Akers;
  + 1> objectGUID: 39ab8618-d3fd-410c-b627-64b65104384d;
  + 1> userAccountControl: 0x200 = ( UF\_NORMAL\_ACCOUNT );
  + 1> codePage: 0;
  + 1> countryCode: 0;
  + 1> pwdLastSet: 07/17/2006 13:50:33 Pacific Standard Daylight Time;
  + 1> primaryGroupID: 513;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1129;
  + 1> accountExpires: 09/14/30828 02:48:05 UNC ;
  + 1> sAMAccountName: KimAkers;
  + 1> sAMAccountType: SAM\_NORMAL\_USER\_ACCOUNT;
  + 1> userPrincipalName: KimAkers@contoso.com;
  + 1> objectCategory: CN=Person, CN=Schema, CN=Configuration, DC=contoso, DC=com;

#### Examples of the IDL\_DRSGetNCChanges Method - Add User to a Group

##### Initial State

User "Kim Akers" is added to the [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) ([[MS-ADSC]](%5bMS-ADSC%5d.pdf#Section_9abb5e97123d4da99557b353ab79b830) section 2.14.) "GroupA" on DC1.

Querying the repsFrom [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) DC=CONTOSO, DC=COM on DC2:

* ldap\_search\_s("DC=contoso,DC=com", *baseObject*, "(objectclass=\*)", *repsFrom*)
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: DC=contoso,DC=com
  + 1> repsFrom: dwVersion = 1,

V1.cb: 276, V1.cConsecutiveFailures: 3 V1.timeLastSuccess: 12797643933 V1.timeLastAttempt: 12797645671 V1.ulResultLastAttempt: 0x2108 V1.cbOtherDraOffset: 216 V1.cbOtherDra: 60 V1.ulReplicaFlags: 0x70

V1.rtSchedule: <ldp:skipped> V1.usnvec.usnHighObjUpdate: 29379 V1.usnvec.usnHighPropUpdate: 29379

V1.uuidDsaObj: c20bc312-4d35-4cc0-9903-b1073368af4a V1.uuidInvocId: c20bc312-4d35-4cc0-9903-b1073368af4a V1.uuidTransportObj: 00000000-0000-0000-0000-000000000000 V1.mtx\_address: c20bc312-4d35-4cc0-9903-b1073368af4a.\_msdcs.contoso.com

V1.cbPASDataOffset: 0 V1.PasData: version = -1, size = -1, flag = -1 ;

Querying the group object "CN=GroupA, CN=Users, DC=CONTOSO, DC=COM" on DC1:

* ldap\_search\_s("CN=GroupA, CN=Users, DC=contoso, DC=com", *baseObject*, "(objectclass=\*)", [*objectClass, cn ... objectCategory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=GroupA,CN=Users,DC=contoso,DC=com
  + 2> objectClass: top; group;
  + 1> cn: GroupA;
  + 2> member: CN=Kim Akers,CN=Users,DC=contoso,DC=com;
  + 1> distinguishedName: CN=GroupA,CN=Users,DC=contoso,DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/13/2006 12:25:35 Pacific Standard Daylight Time;
  + 1> whenChanged: 07/17/2006 14:34:12 Pacific Standard Daylight Time;
  + 1> uSNCreated: 16023;
  + 1> uSNChanged: 29387;
  + 1> name: GroupA;
  + 1> objectGUID: 328ab893-b884-4e31-a73c-71740e261715;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1114;
  + 1> sAMAccountName: GroupA;
  + 1> sAMAccountType: 536870912;
  + 1> groupType: 0x80000004 = ( GROUP\_TYPE\_RESOURCE\_GROUP | GROUP\_TYPE\_SECURITY\_ENABLED );
  + 1> objectCategory: CN=Group,CN=Schema,CN=Configuration,DC=contoso,DC=com;

Querying the [**group object**](#gt_7ce4771c-2043-49b8-85d3-0c60c7789f9a) "CN=GroupA, CN=Users, DC=CONTOSO, DC=COM" on DC2, the member attribute value is not returned, as it is currently empty because this group has no members on DC2.

* ldap\_search\_s("CN=GroupA, CN=Users, DC=contoso, DC=com", *baseObject*, "(objectclass=\*)", [*objectClass, cn ... objectCategory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:

>> Dn: CN=GroupA,CN=Users,DC=contoso,DC=com

* + 2> objectClass: top; group;
  + 1> cn: GroupA;
  + 1> distinguishedName: CN=GroupA,CN=Users,DC=contoso,DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/13/2006 12:25:35 Pacific Standard Daylight Time;
  + 1> whenChanged: 07/13/2006 12:36:03 Pacific Standard Daylight Time;
  + 1> uSNCreated: 26457;
  + 1> uSNChanged: 26543;
  + 1> name: GroupA;
  + 1> objectGUID: 328ab893-b884-4e31-a73c-71740e261715;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1114;
  + 1> sAMAccountName: GroupA;
  + 1> sAMAccountType: 536870912;
  + 1> groupType: 0x80000004 = ( GROUP\_TYPE\_RESOURCE\_GROUP | GROUP\_TYPE\_SECURITY\_ENABLED );
  + 1> objectCategory: CN=Group, CN=Schema, CN=Configuration, DC=contoso, DC=com;

##### Client Request

DC2 invokes the method [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) against DC1, with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 10
* *pmsgIn* = [DRS\_MSG\_GETCHGREQ\_V10](#Section_92b1b77d205846e09e8c6664b96a0cf9)
  + Destination DSA objGuid: \_GUID {6aad8f5a-07cc-403a-9696-9102fe1c320b}
  + Source DSA Invocation ID: \_GUID {c20bc312-4d35-4cc0-9903-b1073368af4a}
  + usnvecFrom: [USN\_VECTOR](#Section_595d11b86ca74a61bd563e6a2b99b76b)
    - usnHighObjUpdate : 29379
    - usnHighPropUpdate : 29379
  + pUpToDateVecDest : [UPTODATE\_VECTOR\_V1\_EXT](#Section_462b424ab50a4c4aa81f48d0f4cf40fe)
    - DSA Invoc ID: c20bc312-4d35-4cc0-9903-b1073368af4a, USN: 29379
  + Flags:
    - DRS\_ASYNC\_OP
    - DRS\_WRIT\_REP
    - DRS\_INIT\_SYNC
    - DRS\_PER\_SYNC
  + Max objects to return: 535
  + Max bytes to return: 5357731
  + Extended operation: none
  + Fsmo Info: 0
  + PrefixTableDest : [SCHEMA\_PREFIX\_TABLE](#Section_9b371267e8b84c69997902dae02e5e38)
  + ulMoreFlags: 0

##### Server Response

Return code of 0 with the following values:

* *pdwOutVersion*= DRS\_MSG\_GETCHGREPLY\_NATIVE\_VERSION\_NUMBER
* *pmsgOut* = [DRS\_MSG\_GETCHGREPLY\_NATIVE](#Section_8079e22efbc04675979cb95cee7f29a5)
  + uuidDsaObjSrc: \_GUID {c20bc312-4d35-4cc0-9903-b1073368af4a}
  + uuidInvocIdSrc: \_GUID {c20bc312-4d35-4cc0-9903-b1073368af4a}
  + pNC: [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) DC=CONTOSO,DC=COM
  + usnvecFrom : [USN\_VECTOR](#Section_595d11b86ca74a61bd563e6a2b99b76b)
    - usnHighObjUpdate : 29379
    - usnHighPropUpdate : 29379
  + usnvecTo: USN\_VECTOR
    - usnHighObjUpdate : 29389
    - usnHighPropUpdate : 29389
  + pUpToDateVecSrc : [UPTODATE\_VECTOR\_V2\_EXT](#Section_cebd1ccb891b4268b0564b714cdf981e)
    - DSA Invoc ID: c20bc312-4d35-4cc0-9903-b1073368af4a
    - usnHighPropUpdate : 29389, timeLastSyncSuccess : 12797646597
  + PrefixTableSrc : [SCHEMA\_PREFIX\_TABLE](#Section_9b371267e8b84c69997902dae02e5e38)
    - pObjects: (null)
  + rgValues: [REPLVALINF\_NATIVE](#Section_882a0aa8fb564be6ad4ab9030314111e)
    - pObject: CN=Kim Akers,CN=Users,DC=contoso,DC=com
    - attrTyp: "member"

##### Final State

Querying the repsFrom [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) DC=CONTOSO, DC=COM on DC2:

* ldap\_search\_s("DC=contoso,DC=com", *baseObject*, "(objectclass=\*)", *repsFrom*)
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: DC=contoso,DC=com

1> repsFrom: dwVersion = 1, V1.cb: 276, V1.cConsecutiveFailures: 0

* + V1.timeLastSuccess: 12797646597 V1.timeLastAttempt: 12797646597 V1.ulResultLastAttempt: 0x0 V1.cbOtherDraOffset: 216
  + V1.cbOtherDra: 60 V1.ulReplicaFlags: 0x70
  + V1.rtSchedule: <ldp:skipped> V1.usnvec.usnHighObjUpdate: 29389
  + V1.usnvec.usnHighPropUpdate: 29389
  + V1.uuidDsaObj: c20bc312-4d35-4cc0-9903-b1073368af4a
  + V1.uuidInvocId: c20bc312-4d35-4cc0-9903-b1073368af4a
  + V1.uuidTransportObj: 00000000-0000-0000-0000-000000000000 V1.mtx\_address: c20bc312-4d35-4cc0-9903-b1073368af4a.\_msdcs.contoso.com
  + V1.cbPASDataOffset: 0 V1.PasData: version = -1, size = -1, flag = -1 ;

Querying the group object "CN=GroupA, CN=Users, DC=CONTOSO, DC=COM" on DC2:

* ldap\_search\_s("CN=GroupA,CN=Users,DC=contoso,DC=com", *baseObject*, "(objectclass=\*)", [*objectClass, cn ... objectCategory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=GroupA,CN=Users,DC=contoso,DC=com
  + 2> objectClass: top; group;
  + 1> cn: GroupA;
  + 1> member: CN=Kim Akers,CN=Users,DC=contoso,DC=com
  + 1> distinguishedName: CN=GroupA,CN=Users,DC=contoso,DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/13/2006 12:25:35 Pacific Standard Daylight Time;
  + 1> whenChanged: 07/17/2006 14:49:46 Pacific Standard Daylight Time;
  + 1> uSNCreated: 26457;
  + 1> uSNChanged: 38218;
  + 1> name: GroupA;
  + 1> objectGUID: 328ab893-b884-4e31-a73c-71740e261715;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1114;
  + 1> sAMAccountName: GroupA;
  + 1> sAMAccountType: 536870912;
  + 1> groupType: 0x80000004 = ( GROUP\_TYPE\_RESOURCE\_GROUP | GROUP\_TYPE\_SECURITY\_ENABLED );
  + 1> objectCategory: CN=Group, CN=Schema, CN=Configuration, DC=contoso, DC=com;

#### Examples of the IDL\_DRSGetNCChanges Method - Change User Password

##### Initial State

User Kim Akers changes the password by pressing CTRL+ALT+DELETE, and the password change is processed by DC1.

Querying the repsFrom [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) DC=CONTOSO, DC=COM on DC2:

* ldap\_search\_s("DC=contoso,DC=com", *baseObject*, "(objectclass=\*)", *repsFrom*)
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: DC=contoso,DC=com
  + 1> repsFrom: dwVersion = 1, V1.cb: 276, V1.cConsecutiveFailures: 4
    - V1.timeLastSuccess: 12797646597 V1.timeLastAttempt: 12797646597 V1.ulResultLastAttempt: 0x2108 V1.cbOtherDraOffset: 216 V1.cbOtherDra: 60 V1.ulReplicaFlags: 0x70
    - V1.rtSchedule: <ldp:skipped> V1.usnvec.usnHighObjUpdate: 29389 V1.usnvec.usnHighPropUpdate: 29389
    - V1.uuidDsaObj: c20bc312-4d35-4cc0-9903-b1073368af4a
    - V1.uuidInvocId: c20bc312-4d35-4cc0-9903-b1073368af4a
    - V1.uuidTransportObj: 00000000-0000-0000-0000-000000000000 V1.mtx\_address: c20bc312-4d35-4cc0-9903-b1073368af4a.\_msdcs.contoso.com
    - V1.cbPASDataOffset: 0 V1.PasData: version = -1, size = -1, flag = -1;

Querying the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) "CN=Kim Akers, CN=Users, DC=CONTOSO,DC=COM" on DC1:

* ldap\_search\_s("CN=Kim Akers,CN=Users,DC=contoso,DC=com", *baseObject*, "(objectClass=\*)", [*objectClass, cn ... objectCategory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=Kim Akers,CN=Users,DC=contoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> cn: Kim Akers;
  + 1> sn: Akers;
  + 1> givenName: Kim;
  + 1> distinguishedName: CN=Kim Akers,CN=Users,DC=contoso,DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/17/2006 13:50:32 Pacific Standard Daylight Time;
  + 1> whenChanged: 07/17/2006 14:58:36 Pacific Standard Daylight Time;
  + 1> displayName: Kim Akers;
  + 1> uSNCreated: 29345;
  + 1> memberOf: CN=GroupA,CN=Users,DC=contoso,DC=com;
  + 1> uSNChanged: 29408;
  + 1> name: Kim Akers;
  + 1> objectGUID: 39ab8618-d3fd-410c-b627-64b65104384d;
  + 1> userAccountControl: 0x200 = ( UF\_NORMAL\_ACCOUNT );
  + 1> badPwdCount: 0;
  + 1> codePage: 0;
  + 1> countryCode: 0;
  + 1> badPasswordTime: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogoff: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogon: 01/01/1601 00:00:00 UNC ;
  + 1> pwdLastSet: 07/17/2006 14:58:36 Pacific Standard Daylight Time;
  + 1> primaryGroupID: 513;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1129;
  + 1> accountExpires: 09/14/30828 02:48:05 UNC ;
  + 1> logonCount: 0;
  + 1> sAMAccountName: KimAkers;
  + 1> sAMAccountType: 805306368;
  + 1> userPrincipalName: KimAkers@contoso.com;
  + 1> objectCategory: CN=Person, CN=Schema, CN=Configuration, DC=contoso, DC=com;

Querying the user object "CN=Kim Akers, CN=Users, DC=CONTOSO,DC=COM" on DC2:

* ldap\_search\_s("CN=Kim Akers,CN=Users,DC=contoso,DC=com", *baseObject*, "(objectclass=\*)", [*objectClass, cn ... objectCategory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=Kim Akers,CN=Users,DC=contoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> cn: Kim Akers;
  + 1> sn: Akers;
  + 1> givenName: Kim;
  + 1> distinguishedName: CN=Kim Akers,CN=Users,DC=contoso,DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/17/2006 13:50:32 Pacific Standard Daylight Time;
  + 1> whenChanged: 07/17/2006 14:05:21 Pacific Standard Daylight Time;
  + 1> displayName: Kim Akers;
  + 1> uSNCreated: 38197;
  + 1> memberOf: CN=GroupA,CN=Users,DC=contoso,DC=com;
  + 1> uSNChanged: 38197;
  + 1> name: Kim Akers;
  + 1> objectGUID: 39ab8618-d3fd-410c-b627-64b65104384d;
  + 1> userAccountControl: 0x200 = ( UF\_NORMAL\_ACCOUNT );
  + 1> codePage: 0;
  + 1> countryCode: 0;
  + 1> pwdLastSet: 07/17/2006 13:50:33 Pacific Standard Daylight Time;
  + 1> primaryGroupID: 513;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1129;
  + 1> accountExpires: 09/14/30828 02:48:05 UNC ;
  + 1> sAMAccountName: KimAkers;
  + 1> sAMAccountType: 805306368;
  + 1> userPrincipalName: KimAkers@contoso.com;
  + 1> objectCategory: CN=Person, CN=Schema, CN=Configuration, DC=contoso,DC=com;

##### Client Request

DC2 invokes the method [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) against DC1, with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 10
* *pmsgIn* = [DRS\_MSG\_GETCHGREQ\_V10](#Section_92b1b77d205846e09e8c6664b96a0cf9)
  + Destination DSA objGuid: \_GUID {6aad8f5a-07cc-403a-9696-9102fe1c320b}
  + Source DSA Invocation ID: \_GUID {c20bc312-4d35-4cc0-9903-b1073368af4a}
  + usnvecFrom: [USN\_VECTOR](#Section_595d11b86ca74a61bd563e6a2b99b76b)
    - usnHighPropUpdate : 29389
    - usnHighObjUpdate : 29389
  + pUpToDateVecDest : [UPTODATE\_VECTOR\_V1\_EXT](#Section_462b424ab50a4c4aa81f48d0f4cf40fe)
    - DSA Invoc ID: c20bc312-4d35-4cc0-9903-b1073368af4a, USN: 29389
  + Flags:
    - DRS\_ASYNC\_OP
    - DRS\_WRIT\_REP
    - DRS\_INIT\_SYNC
    - DRS\_PER\_SYNC
  + Max objects to return: 535
  + Max bytes to return: 5357731
  + Extended operation: none
  + Fsmo Info: 0
  + PrefixTableDest : [SCHEMA\_PREFIX\_TABLE](#Section_9b371267e8b84c69997902dae02e5e38)
  + ulMoreFlags: 0

##### Server Response

A return code of 0 with the following values:

* *pdwOutVersion*= DRS\_MSG\_GETCHGREPLY\_NATIVE\_VERSION\_NUMBER
* *pmsgOut* = [DRS\_MSG\_GETCHGREPLY\_NATIVE](#Section_8079e22efbc04675979cb95cee7f29a5)
  + uuidDsaObjSrc: \_GUID {c20bc312-4d35-4cc0-9903-b1073368af4a}
  + uuidInvocIdSrc: \_GUID {c20bc312-4d35-4cc0-9903-b1073368af4a}
  + pNC: [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) DC=CONTOSO,DC=COM
  + usnvecFrom : [USN\_VECTOR](#Section_595d11b86ca74a61bd563e6a2b99b76b)
    - usnHighObjUpdate : 29389
    - usnHighPropUpdate : 29389
  + usnvecTo : USN\_VECTOR
    - usnHighObjUpdate : 29438
    - usnHighPropUpdate : 29438
  + pUpToDateVecSrc : [UPTODATE\_VECTOR\_V2\_EXT](#Section_cebd1ccb891b4268b0564b714cdf981e)
    - DSA Invoc ID: c20bc312-4d35-4cc0-9903-b1073368af4a,
    - usnHighPropUpdate : 29438, timeLastSyncSuccess : 12797647962
  + PrefixTableSrc : [SCHEMA\_PREFIX\_TABLE](#Section_9b371267e8b84c69997902dae02e5e38)
  + pObjects: [REPLENTINFLIST](#Section_c38b0412cf004b0cb4f44662a4484a00)
    - instanceType: IT\_WRITE
    - dBCSPwd: *binary data*
    - unicodePwd: *binary data*
    - ntPwdHistory: *binary data*
    - pwdLastSet: 07/17/2006 14:58:36 Pacific Standard Daylight Time
    - supplementalCredentials: *binary data*
    - lmPwdHistory: *binary data*

##### Final State

Querying the repsFrom [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) DC=CONTOSO, DC=COM on DC2:

* ldap\_search\_s("DC=contoso,DC=com", *baseObject*, "(objectclass=\*)", *repsFrom*)
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >>Dn: DC=contoso,DC=com
  + 1> repsFrom: dwVersion = 1, V1.cb: 276, V1.cConsecutiveFailures: 0
    - V1.timeLastSuccess: 12797647962 V1.timeLastAttempt: 12797647962 V1.ulResultLastAttempt: 0x0 V1.cbOtherDraOffset: 216 V1.cbOtherDra: 60 V1.ulReplicaFlags: 0x70 V1.rtSchedule: <ldp:skipped> V1.usnvec.usnHighObjUpdate: 29438 V1.usnvec.usnHighPropUpdate: 29438 V1.uuidDsaObj: c20bc312-4d35-4cc0-9903-b1073368af4a
    - V1.uuidInvocId: c20bc312-4d35-4cc0-9903-b1073368af4a
    - V1.uuidTransportObj: 00000000-0000-0000-0000-000000000000 V1.mtx\_address: c20bc312-4d35-4cc0-9903-b1073368af4a.\_msdcs.contoso.com
    - V1.cbPASDataOffset: 0 V1.PasData: version = -1, size = -1, flag = -1;

Querying the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) "CN=Kim Akers, CN=Users, DC=CONTOSO,DC=COM" on DC2, which has now been [**updated**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493):

* ldap\_search\_s("CN=Kim Akers,CN=Users,DC=contoso,DC=com", *baseObject*, "(objectclass=\*)", [*objectClass, cn ... objectCategory*])
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=Kim Akers,CN=Users,DC=contoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> cn: Kim Akers;
  + 1> sn: Akers;
  + 1> givenName: Kim;
  + 1> distinguishedName: CN=Kim Akers,CN=Users,DC=contoso,DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/17/2006 13:50:32 Pacific Standard Daylight Time;
  + 1> whenChanged: 07/17/2006 15:12:35 Pacific Standard Daylight Time;
  + 1> displayName: Kim Akers;
  + 1> uSNCreated: 38197;
  + 1> memberOf: CN=GroupA,CN=Users,DC=contoso,DC=com;
  + 1> uSNChanged: 38270;
  + 1> name: Kim Akers;
  + 1> objectGUID: 39ab8618-d3fd-410c-b627-64b65104384d;
  + 1> userAccountControl: 0x200 = ( UF\_NORMAL\_ACCOUNT );
  + 1> codePage: 0;
  + 1> countryCode: 0;
  + 1> pwdLastSet: 07/17/2006 14:58:36 Pacific Standard Daylight Time;
  + 1> primaryGroupID: 513;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1129;
  + 1> accountExpires: 09/14/30828 02:48:05 UNC ;
  + 1> sAMAccountName: KimAkers;
  + 1> sAMAccountType: 805306368;
  + 1> userPrincipalName: KimAkers@contoso.com;
  + 1> objectCategory: CN=Person, CN=Schema, CN=Configuration, DC=contoso, DC=com;

### IDL\_DRSGetNT4ChangeLog (Opnum 11)

If the server is the [**PDC emulator**](#gt_48b8ecd1-32ae-4593-88e6-346ece75ef34) [**FSMO role owner**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b), the IDL\_DRSGetNT4ChangeLog method returns either a sequence of [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) change log entries or the NT4 [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state, or both, as requested by the client.

1. ULONG IDL\_DRSGetNT4ChangeLog(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_NT4\_CHGLOG\_REQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_NT4\_CHGLOG\_REPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 or ERROR\_MORE\_DATA if successful; another [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if a failure occurred.

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_NT4\_CHGLOG\_REQ

The DRS\_MSG\_NT4\_CHGLOG\_REQ union defines the request messages sent to the [IDL\_DRSGetNT4ChangeLog](#Section_6e000eb660fd4d6cae82bb6479df02fa) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_NT4\_CHGLOG\_REQ\_V1 V1;
6. } DRS\_MSG\_NT4\_CHGLOG\_REQ;

**V1:**  The version 1 request.

##### DRS\_MSG\_NT4\_CHGLOG\_REQ\_V1

The DRS\_MSG\_NT4\_CHGLOG\_REQ\_V1 structure defines the request message sent to the [IDL\_DRSGetNT4ChangeLog](#Section_6e000eb660fd4d6cae82bb6479df02fa) method.

1. typedef struct {
2. DWORD dwFlags;
3. DWORD PreferredMaximumLength;
4. [range(0,10485760)] DWORD cbRestart;
5. [size\_is(cbRestart)] BYTE\* pRestart;
6. } DRS\_MSG\_NT4\_CHGLOG\_REQ\_V1;

**dwFlags:**  Zero or more of the following bit flags, which are presented in little-endian byte order:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | X | X | S N | C L | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**CL (DRS\_NT4\_CHGLOG\_GET\_CHANGE\_LOG, 0x00000001)**: If set, the server returns the [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) change log.

**SN (DRS\_NT4\_CHGLOG\_GET\_SERIAL\_NUMBERS, 0x00000002)**: If set, the server returns the NT4 [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state.

**PreferredMaximumLength:**  The maximum size, in bytes, of the change log data that is to be retrieved in a single operation.

**cbRestart:** Zero if **pRestart** = null. Otherwise, the size, in bytes, of **pRestart**^.

**pRestart:** Null to request the change log from the beginning. Otherwise, a pointer to an opaque value, returned in some previous call to IDL\_DRSGetNT4ChangeLog, identifying the last change log entry returned in that previous call.

##### DRS\_MSG\_NT4\_CHGLOG\_REPLY

The DRS\_MSG\_NT4\_CHGLOG\_REPLY union defines the response messages received from the [IDL\_DRSGetNT4ChangeLog](#Section_6e000eb660fd4d6cae82bb6479df02fa) method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_NT4\_CHGLOG\_REPLY\_V1 V1;
6. } DRS\_MSG\_NT4\_CHGLOG\_REPLY;

**V1:**  The version 1 reply.

##### DRS\_MSG\_NT4\_CHGLOG\_REPLY\_V1

The DRS\_MSG\_NT4\_CHGLOG\_REPLY\_V1 structure defines the response message received from the [IDL\_DRSGetNT4ChangeLog](#Section_6e000eb660fd4d6cae82bb6479df02fa) method.

1. typedef struct {
2. [range(0,10485760)] DWORD cbRestart;
3. [range(0,10485760)] DWORD cbLog;
4. NT4\_REPLICATION\_STATE ReplicationState;
5. DWORD ActualNtStatus;
6. [size\_is(cbRestart)] BYTE\* pRestart;
7. [size\_is(cbLog)] BYTE\* pLog;
8. } DRS\_MSG\_NT4\_CHGLOG\_REPLY\_V1;

**cbRestart:**  Zero if **pRestart** = null. Otherwise, the size, in bytes, of **pRestart**^.

**cbLog:**  Zero if **pRestart** = null. Otherwise, the size, in bytes, of **pLog**^.

**ReplicationState:**  The [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state for Windows NT 4.0 [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**ActualNtStatus:**  A [**STATUS code**](#gt_dfc7ec7a-2b99-4312-a947-5d90a117d1c7). See the pseudo-code for interpretation.

**pRestart:**  Null if no entries were returned. Otherwise, a pointer to an opaque value identifying the last entry returned in **pLog**.

**pLog:**  The buffer containing the next entries from the change log.

##### NT4\_REPLICATION\_STATE

The NT4\_REPLICATION\_STATE structure defines the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state for Windows NT 4.0 [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), whose interpretation is specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.7.1.

1. typedef struct {
2. LARGE\_INTEGER SamSerialNumber;
3. LARGE\_INTEGER SamCreationTime;
4. LARGE\_INTEGER BuiltinSerialNumber;
5. LARGE\_INTEGER BuiltinCreationTime;
6. LARGE\_INTEGER LsaSerialNumber;
7. LARGE\_INTEGER LsaCreationTime;
8. } NT4\_REPLICATION\_STATE;

**SamSerialNumber:**  The Windows NT 4.0 replication [**update sequence number**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) for the SAM database.

**SamCreationTime:**  The time at which the Windows NT 4.0 replication update sequence number for the SAM database was set to 1.

**BuiltinSerialNumber:**  The Windows NT 4.0 replication update sequence number for the built-in database.

**BuiltinCreationTime:**  The time at which the Windows NT 4.0 replication update sequence number for the built-in database was set to 1.

**LsaSerialNumber:**  The Windows NT 4.0 replication update sequence number for the local security authority (LSA) database.

**LsaCreationTime:**  The time at which the Windows NT 4.0 replication update sequence number for the LSA database was set to 1.

#### Method-Specific Abstract Types and Procedures

##### IsPDC

1. procedure IsPDC(): boolean

Returns true if the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) owns the [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) role for this [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), otherwise false.

##### GetWindowsErrorCode

1. procedure GetWindowsErrorCode(ntStatus: DWORD): DWORD

Returns the [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) corresponding to the specified [**STATUS code**](#gt_dfc7ec7a-2b99-4312-a947-5d90a117d1c7).

#### Server Behavior of the IDL\_DRSGetNT4ChangeLog Method

*Informative summary of behavior*: If the server is the [**PDC emulator**](#gt_48b8ecd1-32ae-4593-88e6-346ece75ef34) [**FSMO role owner**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b), it returns either a sequence of [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) change log entries or the NT4 [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state, or both, as requested by the client.

Multiple calls of this method might be required to retrieve the entire PDC change log. The client passes pRestart = null on the first call in a series of calls; the server returns a sequence of change log entries, including the first, a pointer to an opaque cookie, and a result code. If the server returns no change log entries, it returns null instead of a pointer to a cookie. If the server returns the result code zero, the sequence of change log entries in the response includes the final entry in the log.

The cookie encodes the serial number of the last change log entry returned. If the server returns ERROR\_MORE\_DATA, the final change log entry in the response was not the final entry in the change log. The client can make another call, with pRestart pointing to the cookie. The server processes this call identically to a call with pRestart = null, except that it returns change log entries starting with the entry following the last previously returned entry, as indicated by the cookie. By making enough calls the client can retrieve the entire change log.

If the client includes a cookie that is either corrupted or identifies a nonexistent change log entry (possibly because the cookie is too old), the server returns ERROR\_INVALID\_PARAMETER. If there are change log entries to return, but the client specifies a bound on the size of the returned change log entries that is too small to hold even a single entry, the server returns ERROR\_INSUFFICIENT\_BUFFER.

The NT4 replication state is a small, fixed-size structure and the server simply copies it into the response.

When the client requests both the PDC change log and the NT4 replication state, the server processes the PDC change log request first. If an error occurs during this processing the server does not process the request for NT4 replication state. If an error occurs while processing the NT4 replication state request, the server returns no indication to the client that the PDC change log request succeeded.

1. ULONG
2. IDL\_DRSGetNT4ChangeLog(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_NT4\_CHGLOG\_REQ \*pmsgIn,
6. [out, ref] DWORD \*pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_NT4\_CHGLOG\_REPLY \*pmsgOut)
9. msgIn: DRS\_MSG\_NT4\_CHGLOG\_REQ\_V1
10. readStatus, ntStatus: DWORD
11. sequenceNumber: integer
12. nextIndexToBeReturned, lastIndexToBeReturned: integer
13. lastReturnedSerialNumber: LONGLONG
14. lastReturnedIndex: integer
15. pChangeLog: ADDRESS OF CHANGE\_LOG\_ENTRIES
16. ValidateDRSInput(hDrs, 11)
17. pdwOutVersion^ := 1
18. pmsgOut^.V1.cbRestart := 0
19. pmsgOut^.V1.cbLog := 0
20. pmsgOut^.V1.ReplicationState.SamSerialNumber := 0
21. pmsgOut^.V1.ReplicationState.SamCreationTime := 0
22. pmsgOut^.V1.ReplicationState.BuiltinSerialNumber := 0
23. pmsgOut^.V1.ReplicationState.BuiltinCreationTime := 0
24. pmsgOut^.V1.ReplicationState.LsaSerialNumber := 0
25. pmsgOut^.V1.ReplicationState.LsaCreationTime := 0
26. pmsgOut^.V1.ActualNtStatus := 0
27. pmsgOut^.V1.pRestart := null
28. pmsgOut^.V1.pLog := null
29. /\* Validate the request version \*/
30. if dwInVersion ≠ 1 then
31. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
32. endif
33. msgIn := pmsgIn^.V1
34. /\* Access check \*/
35. if not AccessCheckCAR(DefaultNC(), DS-Replication-Get-Changes) then
36. return ERR0R\_ACCESS\_DENIED
37. endif
38. /\* The DC must own the PDC role \*/
39. if not IsPDC() then
40. return ERROR\_INVALID\_DOMAIN\_ROLE
41. endif
42. ntStatus := 0
43. readStatus := 0
44. if DRS\_NT4\_CHGLOG\_GET\_CHANGE\_LOG in msgIn.dwFlags then
45. /\* Return NT4 change log entries. \*/
46. /\* Determine the position of the first entry in the change log that
47. \* needs to be returned. If pRestart = null, this is the first
48. \* entry of the change log, otherwise it is the entry following the
49. \* entry identified in the cookie pRestart^. \*/
50. if msgIn.pRestart = null then
51. sequenceNumber := 1
52. nextIndexToBeReturned := 0
53. else
54. sequenceNumber :=
55. (Sequence number extracted from msgIn.pRestart^) + 1
56. lastReturnedSerialNumber :=
57. Serial number extracted from msgIn.pRestart^
58. lastReturnedIndex := select one i in dc.pdcChangeLog where
59. dc.pdcChangeLog[i].SerialNumber = lastReturnedSerialNumber
60. if lastReturnedIndex = null then
61. /\* Cookie is old or corrupted.
62. \* The STATUS code STATUS\_INVALID\_PARAMETER corresponds to
63. \* the Windows error code ERROR\_INVALID\_PARAMETER. \*/
64. ntStatus := STATUS\_INVALID\_PARAMETER
65. else
66. nextIndexToBeReturned := lastReturnedIndex + 1
67. endif
68. endif
69. if ntStatus = 0 and nextIndexToBeReturned ≥ dc.pdcChangeLog.length
70. then
71. /\* No entries to be returned, complete the response message \*/
72. pmsgOut^.V1.pLog := null
73. pmsgOut^.V1.cbLog := 0
74. pmsgOut^.V1.pRestart := null
75. pmsgOut^.V1.cbRestart := 0
76. endif
77. if ntStatus = 0 and
78. nextIndexToBeReturned < dc.pdcChangeLog.length then
79. /\* Entries to be returned. First, determine how many entries fit
80. \* into the response message \*/
81. lastIndexToBeReturned := the largest integer q such that
82. q < dc.pdcChangeLog.length and
83. the size in bytes of
84. dc.pdcChangeLog[nextIndexToBeReturned .. q]
85. is <= msgIn.PreferredMaximumLength
86. if lastIndexToBeReturned < nextIndexToBeReturned then
87. /\* Client's PreferredMaximumLength is too small for a single
88. \* entry, so return no entries.
89. \* The STATUS code STATUS\_BUFFER\_TOO\_SMALL corresponds to
90. \* the Windows error code ERROR\_INSUFFICIENT\_BUFFER. \*/
91. ntStatus := STATUS\_BUFFER\_TOO\_SMALL
92. else
93. /\* Client's PreferredMaximumLength is large enough for one or
94. \* more entries. Fill in pChangeLog^ from dc.pdcChangeLog \*/
95. pChangeLog^.Size := 0x00000010
96. pChangeLog^.Version := 0x00000001
97. pChangeLog^.SequenceNumber := sequenceNumber
98. pChangeLog^.Flags := 0x00000000
99. pChangeLog^.ChangeLogEntries :=
100. dc.pdcChangeLog[nextIndexToBeReturned ..
101. lastIndexToBeReturned]
102. if a fatal error occurred while retrieving dc.pdcChangeLog then
103. ntStatus :=
104. STATUS code of error that occurred, high-order bit set
105. end
106. endif
107. if ntStatus = 0 then
108. /\* No errors, complete the response message \*/
109. pmsgOut^.V1.pLog := pChangeLog
110. pmsgOut^.V1.cbLog := size in bytes of pmsgOut^.V1.pLog^
111. /\* Construct a new cookie \*/
112. lastReturnedSerialNumber :=
113. dc.pdcChangeLog[lastIndexToBeReturned].SerialNumber
114. pmsgOut^.V1.pRestart :=
115. ADDRESS OF implementation-specific struct
116. encapsulating lastReturnedSerialNumber and sequenceNumber
117. pmsgOut^.V1.cbRestart := size in bytes of pmsgOut^.V1.pRestart^
118. if lastIndexToBeReturned < dc.pdcChangeLog.length - 1 then
119. /\* There are more entries to be returned.
120. \* The STATUS code STATUS\_MORE\_ENTRIES corresponds to
121. \* the Windows error code ERROR\_MORE\_DATA. \*/
122. ntStatus := STATUS\_MORE\_ENTRIES
123. endif
124. endif /\* Response complete \*/
125. endif /\* Entries returned \*/
126. endif /\* Processed change log request \*/
127. /\* Save the status code from the previous operation \*/
128. readStatus := ntStatus
129. if ntStatus < 0x80000000 and
130. DRS\_NT4\_CHGLOG\_GET\_SERIAL\_NUMBERS in msgIn.dwFlags then
131. /\* Return NT4 replication state. \*/
132. pmsgOut^.V1.ReplicationState.SamSerialNumber :=
133. dc.nt4ReplicationState.SamNT4ReplicationUSN
134. pmsgOut^.V1.ReplicationState.SamCreationTime :=
135. dc.nt4ReplicationState.SamCreationTime
136. pmsgOut^.V1.ReplicationState.BuiltinSerialNumber :=
137. dc.nt4ReplicationState.BuiltinNT4ReplicationUSN
138. pmsgOut^.V1.ReplicationState.BuiltinCreationTime :=
139. dc.nt4ReplicationState.BuiltinCreationTime
140. pmsgOut^.V1.ReplicationState.LsaSerialNumber := 1
141. pmsgOut^.V1.ReplicationState.LsaCreationTime :=
142. current time on the DC
143. if a fatal error occurred while retrieving NT4 replication state
144. then
145. ntStatus :=
146. STATUS code of error that occurred, high-order bit set
147. end
148. endif
149. if ntStatus < 0x80000000 then
150. pmsgOut^.V1.ActualStatus := readStatus
151. else
152. pmsgOut^.V1.ActualStatus := ntStatus
153. endif
154. return GetWindowsErrorCode(ntStatus)

#### Examples of the IDL\_DRSGetNT4ChangeLog Method

##### Initial State

[**Domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) functional level is DS\_BEHAVIOR\_WIN2000 and the nTMixedDomain [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) root is 1 (see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.4.1). The [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) role is held by DC2.

* ldap\_search\_s("DC=contoso,DC=com", *baseObject*, "(objectClass=\*)")
* >> Dn: DC=contoso,DC=com
  + 1> fSMORoleOwner: CN=NTDS Settings, CN=DC2, CN=Servers,

CN=Default-First-Site-Name, CN=Sites, CN=Configuration,

DC=contoso, DC=com;

##### Client Request

The [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) role is transferred to DC1, which results in DC1 invoking the [IDL\_DRSGetNT4ChangeLog](#Section_6e000eb660fd4d6cae82bb6479df02fa) method against DC2 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC2 omitted):

1. dwMsgVersion = 1
2. pmsgIn =
3. dwFlags: DRS\_NT4\_CHGLOG\_GET\_CHANGE\_LOG +
4. DRS\_NT4\_CHGLOG\_GET\_SERIAL\_NUMBERS
5. PreferredMaximumLength: 0x4000
6. cbRestart: 0
7. pRestart: null

##### Server Response

Return code of 0 with the following values:

1. pmsgOut = DRS\_MSG\_NT4\_CHGLOG\_REPLY\_V1
2. cbRestart = 0x10
3. cbLog = 0x2d00
4. ReplicationState = \_NT4\_REPLICATION\_STATE
5. SamSerialNumber = 0x30`00000097
6. SamCreationTime = 0x1c6a7a9`792f51f6
7. BuiltinSerialNumber = 0x30`00000054
8. BuiltinCreationTime = 0x1c6a7a9`792f51f6
9. LsaSerialNumber = 0x1
10. LsaCreationTime = 0x1c6a832`0a495151
11. ActualNtStatus = 0
12. pRestart = "LMEM"
13. pLog = pointer to actual log (log data omitted)

##### Final State

The [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) change log entries on DC1 are synchronized with the change log entries on DC2; there is no other change in state.

### IDL\_DRSGetObjectExistence (Opnum 23)

The IDL\_DRSGetObjectExistence method helps the client check the consistency of [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) existence between its [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) of an [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) and the server's replica of the same NC. Checking the consistency of object existence means identifying objects that have replicated to both replicas and that exist in one replica but not in the other. For the purposes of this method, an object *exists* within a [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) if it is either an object or a [**tombstone**](#gt_9d8e0963-13fa-4e19-a97f-7ce6bc90d20f).

See [IDL\_DRSReplicaVerifyObjects](#Section_8dba150d50f647f1941e1a606c30db0b) for a use of this method.

1. ULONG IDL\_DRSGetObjectExistence(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_EXISTREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_EXISTREPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_EXISTREQ

The DRS\_MSG\_EXISTREQ union defines request messages sent to the [IDL\_DRSGetObjectExistence](#Section_6355d4f5f5564527adde37afba2fcf56) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_EXISTREQ\_V1 V1;
6. } DRS\_MSG\_EXISTREQ;

**V1:**  The version 1 request.

##### DRS\_MSG\_EXISTREQ\_V1

The DRS\_MSG\_EXISTREQ\_V1 structure defines the request message sent to the [IDL\_DRSGetObjectExistence](#Section_6355d4f5f5564527adde37afba2fcf56) method.

1. typedef struct {
2. UUID guidStart;
3. DWORD cGuids;
4. DSNAME\* pNC;
5. UPTODATE\_VECTOR\_V1\_EXT\* pUpToDateVecCommonV1;
6. UCHAR Md5Digest[16];
7. } DRS\_MSG\_EXISTREQ\_V1;

**guidStart:**  The objectGUID of the first [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the client's object sequence.

**cGuids:**  The number of objects in the client's object sequence.

**pNC:**  The [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) containing the objects in the sequence.

**pUpToDateVecCommonV1:**  The filter excluding objects from the client's object sequence.

**Md5Digest:**  The [**digest**](#gt_1c222b9e-7176-4840-9d19-e65895b9fc62) of the objectGUID values of the objects in the client's object sequence.

##### DRS\_MSG\_EXISTREPLY

The DRS\_MSG\_EXISTREPLY union defines the response message versions received from the [IDL\_DRSGetObjectExistence](#Section_6355d4f5f5564527adde37afba2fcf56) method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_EXISTREPLY\_V1 V1;
6. } DRS\_MSG\_EXISTREPLY;

**V1:**  The version 1 response.

##### DRS\_MSG\_EXISTREPLY\_V1

The DRS\_MSG\_EXISTREPLY\_V1 structure defines the response message received from the [IDL\_DRSGetObjectExistence](#Section_6355d4f5f5564527adde37afba2fcf56) method.

1. typedef struct {
2. DWORD dwStatusFlags;
3. [range(0,10485760)] DWORD cNumGuids;
4. [size\_is(cNumGuids)] UUID\* rgGuids;
5. } DRS\_MSG\_EXISTREPLY\_V1;

**dwStatusFlags:**  1 if the [**digests**](#gt_1c222b9e-7176-4840-9d19-e65895b9fc62) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) sequences on the client and server are the same, 0 if they are different.

**cNumGuids:**  The number of items in the **rgGuids** array. Zero if **dwStatusFlags** = 0.

**rgGuids:**   The objectGUIDs of the objects in the server's object sequence.

#### Method-Specific Abstract Types and Procedures

The following procedure is used in specifying both the client and server behavior of [IDL\_DRSGetObjectExistence](#Section_6355d4f5f5564527adde37afba2fcf56).

##### GuidSequence

*Informative summary of behavior*: The *candidate set* of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) is the set of all objects and [**tombstones**](#gt_9d8e0963-13fa-4e19-a97f-7ce6bc90d20f) in the local [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) of [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) *nc*, excluding objects that have not yet replicated to both the client and server replicas of NC *nc*. The exclusion of objects created too recently is performed using the client-supplied [**up-to-date vector**](#gt_42564a26-2ae7-41a2-a67c-3c74381d8538) *utd*.

A *cluster* is any subset of the candidate set such that no object in the candidate set outside the cluster has an objectGUID lying between the objectGUIDs of any two members of the cluster.

The cluster constructed by GuidSequence contains the object in the candidate set with the smallest objectGUID greater than or equal to *startGUID*. The cluster contains as many objects as possible, but no more than count.

Both the client and the server use GuidSequence to compute a cluster, create a sorted sequence of objectGUIDs of objects in the cluster, and compute a [**digest**](#gt_1c222b9e-7176-4840-9d19-e65895b9fc62) of that sequence.

1. procedure GuidSequence(
2. startGUID: GUID,
3. count: ULONG,
4. nc: DSName,
5. utd: UPTODATE\_VECTOR\_V1\_EXT,
6. var s: sequence of DSName,
7. var digest: sequence [0..15] of byte)

The procedure GuidSequence returns the following:

1. A sequence *s* of objectGUIDs from the server's state.
2. An MD5 digest value *digest* that is derived from the sequence *s*.

The first four parameters determine the result sequence *s* as follows:

1. Construct the following set of [DSName](#Section_a0d5477a522946b9890a54b924d487d1)s:
2. select all o subtree-ts-included nc where
3. StampLessThanOrEqualUTD(AttrStamp(o, whenCreated), utd)
4. Construct the [GUID](#Section_5e740f50e6a048c9bca800072e85d963) sequence *S* that contains the objectGUIDs of members of the set, sorted into ascending order by GUID value.
5. Find the smallest integer *i* such that *S*[*i*] >= *startGUID*. If there is no such *i*, the result sequence *s* is empty, otherwise the result sequence *s* is as follows:

*S*[*i* .. min(*i*+count, *S*.length)-1]

The result digest is the value of ComputeDigest applied to the result sequence *s*, where ComputeDigest is specified as follows:

1. procedure ComputeDigest(s: sequence of GUID): sequence [0..15] of byte
2. md5Context : MD5\_CTX
3. MD5Init(md5Context)
4. for i := 0 to s.length-1
5. MD5Update(md5Context, s[i], 16)
6. endfor
7. MD5Final(md5Context)
8. return md5Context.digest

#### Client Behavior When Sending the IDL\_DRSGetObjectExistence Request

*Informative summary of behavior*: The client uses [IDL\_DRSGetObjectExistence](#Section_6355d4f5f5564527adde37afba2fcf56) to check the consistency of [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) existence between its [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) of an [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) and another replica of the same NC. Checking the consistency of object existence means identifying objects that have replicated to both replicas, and that exist in one replica but not in the other. For the purposes of this method, an object *exists* within an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) if it is either an object or a [**tombstone**](#gt_9d8e0963-13fa-4e19-a97f-7ce6bc90d20f).

IDL\_DRSGetObjectExistence allows the client to perform this checking in *clusters*, as defined in the informative summary of the GuidSequence procedure (section [4.1.12.2.1](#Section_fd7a7308e38b42c880b24f4f7d143c4d)).

The inputs to this checking process on the client are as follows:

1. nc: DSName
2. utdClient, utdServer: UPTODATE\_VECTOR\_V1\_EXT
3. guidStart: GUID
4. count: ULONG

**nc**: The NC containing the cluster that the client will check.

**utdClient, utdServer**: The [**up-to-date vectors**](#gt_42564a26-2ae7-41a2-a67c-3c74381d8538) of the client and server for the NC *nc*, respectively. The client can obtain *utdServer* using [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47).

**guidStart**: The lower bound on the smallest objectGUID in the cluster that the client will check.

**count**: The upper bound on the number of objects in the cluster that the client will check.

Given these inputs, the client creates the request message to IDL\_DRSGetObjectExistence as follows:[<32>](#Appendix_A_32" \o "Product behavior note 32)

1. msgIn: DRS\_MSG\_EXISTREQ\_V1
2. s: sequence
3. digest: sequence [0..15] of byte
4. msgIn.pNC := nc
5. msgIn.pUpToDateVecCommonV1 := MergeUTD(utdClient, utdServer)
6. GuidSequence(
7. guidStart, count, nc, msgIn.pUpToDateVecCommonV1^, s, digest)
8. msgIn.guidStart := s[0]
9. msgIn.length := s.length
10. msgIn.Md5Digest := digest
11. pmsgIn^.V1 := msgIn

#### Server Behavior of the IDL\_DRSGetObjectExistence Method

*Informative summary of behavior*: The server computes a cluster, an objectGUID sequence, and a [**digest**](#gt_1c222b9e-7176-4840-9d19-e65895b9fc62) in the same manner as the client, but uses the server's [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). If the digest computed by the server equals the digest in the client's request, the server returns dwStatusFlags = 1, otherwise the server returns dwStatusFlags = 0 and the objectGUID sequence.

1. ULONG IDL\_DRSGetObjectExistence (
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_EXISTREQ \*pmsgIn,
5. [out, ref] DWORD \*pdwOutVersion,
6. [out, ref, switch\_is(\*pdwOutVersion)]
7. DRS\_MSG\_EXISTREPLY \*pmsgOut)
8. msgIn: DRS\_MSG\_EXISTREQ\_V1
9. nc: DSName
10. s: sequence of GUID
11. digest: sequence [0..15] of byte
12. msgOut: DRS\_MSG\_EXISTREPLY\_V1
13. \*pdwOutVersion = 1;
14. ValidateDRSInput(hDrs, 23)
15. pdwOutVersion^ := 1
16. pmsgOut^.V1.dwStatusFlags := 0
17. pmsgOut^.V1.cNumGuids := 0
18. pmsgOut^.V1.rgGuids := null
19. if dwInVersion ≠ 0x1 then
20. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
21. endif
22. msgIn := pmsgIn^.V1
23. nc := msgIn.pNC^
24. if not MasterReplicaExists(nc) then
25. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
26. endif
27. if msgIn.guidStart = NULLGUID then
28. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
29. endif
30. if not AccessCheckCAR(nc, DS-Replication-Get-Changes) then
31. return ERROR\_DS\_DRA\_ACCESS\_DENIED
32. endif
33. GuidSequence(msgIn.guidStart, msgIn.cGuids, nc,
34. msgIn.pUpToDateVecCommonV1^, s, digest)
35. if msgIn.Md5Digest = digest then
36. msgOut.dwStatusFlags := 1
37. msgOut.cNumGuids := 0
38. msgOut.rgGuids := null
39. else if
40. msgOut.dwStatusFlags := 0
41. msgOut.cNumGuids := s.length
42. for i := 0 to s.length - 1
43. msgOut.rgGuids[i] := s[i]
44. endfor
45. endif
46. pmsgOut^.V1 := msgOut
47. return 0

#### Client Behavior When Receiving the IDL\_DRSGetObjectExistence Response

*Informative summary of behavior*: If the server response contains dwStatusFlags = 0, the client computes the difference between the client and the server sequences and takes whatever action is required.

### IDL\_DRSGetReplInfo (Opnum 19)

The IDL\_DRSGetReplInfo method retrieves the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state of the server.

1. ULONG IDL\_DRSGetReplInfo(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_GETREPLINFO\_REQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_GETREPLINFO\_REPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_GETREPLINFO\_REQ

The DRS\_MSG\_GETREPLINFO\_REQ union defines the request message versions sent to the [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_GETREPLINFO\_REQ\_V1 V1;
6. [case(2)]
7. DRS\_MSG\_GETREPLINFO\_REQ\_V2 V2;
8. } DRS\_MSG\_GETREPLINFO\_REQ;

**V1:**  Version 1 request.

**V2:**  Version 2 request. The V2 request structure is a superset of the V1 request structure.

##### DRS\_MSG\_GETREPLINFO\_REQ\_V1

The DRS\_MSG\_GETREPLINFO\_REQ\_V1 structure defines a version 1 request message sent to the [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47) method.

1. typedef struct {
2. DWORD InfoType;
3. [string] LPWSTR pszObjectDN;
4. UUID uuidSourceDsaObjGuid;
5. } DRS\_MSG\_GETREPLINFO\_REQ\_V1;

**InfoType:**  MUST be a DS\_REPL\_INFO code.

**pszObjectDN:**  [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) on which the operation is to be performed. The meaning of this parameter depends on the value of the **InfoType** parameter.

**uuidSourceDsaObjGuid:**  [**NULL GUID**](#gt_ba500a5b-8c29-467c-a335-0980c8b11304) or the [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

##### DRS\_MSG\_GETREPLINFO\_REQ\_V2

The DRS\_MSG\_GETREPLINFO\_REQ\_V2 structure defines a version 2 request message sent to the [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47) method. The V2 request structure is a superset of the V1 request structure.

1. typedef struct {
2. DWORD InfoType;
3. [string] LPWSTR pszObjectDN;
4. UUID uuidSourceDsaObjGuid;
5. DWORD ulFlags;
6. [string] LPWSTR pszAttributeName;
7. [string] LPWSTR pszValueDN;
8. DWORD dwEnumerationContext;
9. } DRS\_MSG\_GETREPLINFO\_REQ\_V2;

**InfoType:**  MUST be a DS\_REPL\_INFO code.

**pszObjectDN:**  [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) on which the operation is to be performed. The meaning of this parameter depends on the value of the **InfoType** parameter.

**uuidSourceDsaObjGuid:**  [**NULL GUID**](#gt_ba500a5b-8c29-467c-a335-0980c8b11304) or the [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**ulFlags:**  Zero or more of the following bit flags, which are presented in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | X | X | X | M T | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**MT (DS\_REPL\_INFO\_FLAG\_IMPROVE\_LINKED\_ATTRS, 0x00000001)**: Return [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) [**stamps**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) for linked values.

**pszAttributeName:**  Null, or the lDAPDisplayName of a [**link attribute**](#gt_be41074d-ce6b-4488-853a-4bbb3ea243ce).

**pszValueDN:**  Null, or the DN of the [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) for which to retrieve a stamp.

**dwEnumerationContext:**  Zero, or the value of **dwEnumerationContext** returned by the server on a previous call to IDL\_DRSGetReplInfo. For an informative description of the sequencing issues associated with this field, see section [1.3.2](#Section_67c5a415a6c740988cf36ef8d173cfe8).

##### DS\_REPL\_INFO Codes

DS\_REPL\_INFO codes indicate the type of [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state information being requested.

| Value | Meaning |
| --- | --- |
| DS\_REPL\_INFO\_NEIGHBORS  0x00000000 | Replication state data for each [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) and source server pair, for all [**NC replicas**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) hosted by this [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). |
| DS\_REPL\_INFO\_CURSORS\_FOR\_NC  0x00000001 | A portion of the replication state for the NC replica of a given NC. |
| DS\_REPL\_INFO\_METADATA\_FOR\_OBJ  0x00000002 | [**Stamps**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) for all the [**replicated attributes**](#gt_74537956-c9e2-414a-a684-0fd75f204181) of the given [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). |
| DS\_REPL\_INFO\_KCC\_DSA\_CONNECT\_FAILURES  0x00000003 | Replication state data regarding connection failures with inbound replication partners. |
| DS\_REPL\_INFO\_KCC\_DSA\_LINK\_FAILURES  0x00000004 | Replication state data regarding link failures with inbound replication partners. |
| DS\_REPL\_INFO\_PENDING\_OPS  0x00000005 | Replication tasks that are currently executing or that are queued to execute. |
| DS\_REPL\_INFO\_METADATA\_FOR\_ATTR\_VALUE  0x00000006 | Stamps for a specific [**link attribute**](#gt_be41074d-ce6b-4488-853a-4bbb3ea243ce) of the given object. |
| DS\_REPL\_INFO\_CURSORS\_2\_FOR\_NC  0x00000007 | A portion of the replication state for the NC replica of a given NC. |
| DS\_REPL\_INFO\_CURSORS\_3\_FOR\_NC  0x00000008 | A portion of the replication state for the NC replica of a given NC. |
| DS\_REPL\_INFO\_METADATA\_2\_FOR\_OBJ  0x00000009 | Stamps for all the replicated attributes of the given object. |
| DS\_REPL\_INFO\_METADATA\_2\_FOR\_ATTR\_VALUE  0x0000000A | Stamps for a specific link attribute of the given object. |
| DS\_REPL\_INFO\_SERVER\_OUTGOING\_CALLS  0xFFFFFFFA | A list of all outstanding [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) server call contexts. |
| DS\_REPL\_INFO\_UPTODATE\_VECTOR\_V1  0xFFFFFFFB | The replication state for the NC replica of a given NC. |
| DS\_REPL\_INFO\_CLIENT\_CONTEXTS  0xFFFFFFFC | A list of all outstanding RPC client contexts. |
| DS\_REPL\_INFO\_REPSTO  0xFFFFFFFE | Replication state data for each NC and destination server (which is notified of changes) pair, for all NC replicas hosted by this DC. |

##### DRS\_MSG\_GETREPLINFO\_REPLY

The DRS\_MSG\_GETREPLINFO\_REPLY union defines response messages received from the [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(0)]
5. DS\_REPL\_NEIGHBORSW\* pNeighbors;
6. [case(1)]
7. DS\_REPL\_CURSORS\* pCursors;
8. [case(2)]
9. DS\_REPL\_OBJ\_META\_DATA\* pObjMetaData;
10. [case(3)]
11. DS\_REPL\_KCC\_DSA\_FAILURESW\* pConnectFailures;
12. [case(4)]
13. DS\_REPL\_KCC\_DSA\_FAILURESW\* pLinkFailures;
14. [case(5)]
15. DS\_REPL\_PENDING\_OPSW\* pPendingOps;
16. [case(6)]
17. DS\_REPL\_ATTR\_VALUE\_META\_DATA\* pAttrValueMetaData;
18. [case(7)]
19. DS\_REPL\_CURSORS\_2\* pCursors2;
20. [case(8)]
21. DS\_REPL\_CURSORS\_3W\* pCursors3;
22. [case(9)]
23. DS\_REPL\_OBJ\_META\_DATA\_2\* pObjMetaData2;
24. [case(10)]
25. DS\_REPL\_ATTR\_VALUE\_META\_DATA\_2\* pAttrValueMetaData2;
26. [case(0xFFFFFFFA)]
27. DS\_REPL\_SERVER\_OUTGOING\_CALLS\* pServerOutgoingCalls;
28. [case(0xFFFFFFFB)]
29. UPTODATE\_VECTOR\_V1\_EXT\* pUpToDateVec;
30. [case(0xFFFFFFFC)]
31. DS\_REPL\_CLIENT\_CONTEXTS\* pClientContexts;
32. [case(0xFFFFFFFE)]
33. DS\_REPL\_NEIGHBORSW\* pRepsTo;
34. } DRS\_MSG\_GETREPLINFO\_REPLY;

**pNeighbors:**  Neighbor information.

**pCursors:**  Cursors for an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

**pObjMetaData:**  [**Attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) [**stamps**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00).

**pConnectFailures:**  Connection failure data.

**pLinkFailures:**  Link failure data.

**pPendingOps:**  Pending operations in the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) queue.

**pAttrValueMetaData:**  [**Link value stamps**](#gt_8878eaba-14b8-463b-94be-5784c1d24a85).

**pCursors2:**  Cursors for an NC replica.

**pCursors3:**  Cursors for an NC replica.

**pObjMetaData2:**  Attribute stamps.

**pAttrValueMetaData2:**  Link value stamps.

**pServerOutgoingCalls:**  Outstanding requests from this [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) to other DCs.

**pUpToDateVec:**  Cursors for an NC replica.

**pClientContexts:**  Active [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) client connections.

**pRepsTo:**  Neighbor information.

##### DS\_REPL\_NEIGHBORSW

The DS\_REPL\_NEIGHBORSW structure defines a set of [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) neighbors. This structure is a concrete representation of a sequence of [RepsFrom](#Section_3ef27d3cb9c944048e53ebf3a64a9a10) or [RepsTo](#Section_302391a9f6e14c0ca1b25604a42e982b) values.

1. typedef struct {
2. DWORD cNumNeighbors;
3. DWORD dwReserved;
4. [size\_is(cNumNeighbors)] DS\_REPL\_NEIGHBORW rgNeighbor[];
5. } DS\_REPL\_NEIGHBORSW;

**cNumNeighbors:**  The count of items in the **rgNeighbor** array.

**dwReserved:**  Unused. MUST be 0 and ignored.

**rgNeighbor:**  A set of replication neighbors.

##### DS\_REPL\_NEIGHBORW

The DS\_REPL\_NEIGHBORW structure defines a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) neighbor. This structure is a concrete representation of a [RepsFrom](#Section_3ef27d3cb9c944048e53ebf3a64a9a10) or [RepsTo](#Section_302391a9f6e14c0ca1b25604a42e982b) value.

1. typedef struct {
2. [string] LPWSTR pszNamingContext;
3. [string] LPWSTR pszSourceDsaDN;
4. [string] LPWSTR pszSourceDsaAddress;
5. [string] LPWSTR pszAsyncIntersiteTransportDN;
6. DWORD dwReplicaFlags;
7. DWORD dwReserved;
8. UUID uuidNamingContextObjGuid;
9. UUID uuidSourceDsaObjGuid;
10. UUID uuidSourceDsaInvocationID;
11. UUID uuidAsyncIntersiteTransportObjGuid;
12. USN usnLastObjChangeSynced;
13. USN usnAttributeFilter;
14. FILETIME ftimeLastSyncSuccess;
15. FILETIME ftimeLastSyncAttempt;
16. DWORD dwLastSyncResult;
17. DWORD cNumConsecutiveSyncFailures;
18. } DS\_REPL\_NEIGHBORW;

**pszNamingContext:**  The [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

**pszSourceDsaDN:**  The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the server [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

**pszSourceDsaAddress:**  The [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) of the server DC.

**pszAsyncIntersiteTransportDN:**  The DN of the interSiteTransport object corresponding to the transport used to communicate with the server DC.

**dwReplicaFlags:**  The [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) flags.

**dwReserved:**  Unused. MUST be 0 and ignored.

**uuidNamingContextObjGuid:**  The objectGUID of the NC root.

**uuidSourceDsaObjGuid:**  The [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the server DC.

**uuidSourceDsaInvocationID:**  The [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) associated with the server DC.

**uuidAsyncIntersiteTransportObjGuid:**  The objectGUID of the interSiteTransport object corresponding to the transport used to communicate with the server DC.

**usnLastObjChangeSynced:**  An implementation-specific value.

**usnAttributeFilter:**  An implementation-specific value.

**ftimeLastSyncSuccess:**  The time of the last successful replication from the server DC.

**ftimeLastSyncAttempt:**  The time of the last attempt to replicate from the server DC.

**dwLastSyncResult:**  0, or the [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b), as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.2, resulting from the last sync attempt.

**cNumConsecutiveSyncFailures:**  The number of consecutive failures to replicate from the server DC.

##### DS\_REPL\_CURSORS

The DS\_REPL\_CURSORS structure defines a set of [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) cursors for a given [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). This structure is a concrete representation of a sequence of [ReplUpToDateVector](#Section_8cb40d62a51d47e39b4e0837edffd61c) values.

1. typedef struct {
2. DWORD cNumCursors;
3. DWORD dwReserved;
4. [size\_is(cNumCursors)] DS\_REPL\_CURSOR rgCursor[];
5. } DS\_REPL\_CURSORS;

**cNumCursors:**  The count of items in the **rgCursor** array.

**dwReserved:**  Unused. MUST be 0 and ignored.

**rgCursor:**  A set of replication cursors.

##### DS\_REPL\_CURSOR

The DS\_REPL\_CURSOR structure defines a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) cursor for a given [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). This structure is a concrete representation of a [ReplUpToDateVector](#Section_8cb40d62a51d47e39b4e0837edffd61c) value.

1. typedef struct {
2. UUID uuidSourceDsaInvocationID;
3. USN usnAttributeFilter;
4. } DS\_REPL\_CURSOR;

**uuidSourceDsaInvocationID:**  The [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**usnAttributeFilter:**  The [**update sequence number (USN)**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) at which an [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) was applied on the DC.

##### DS\_REPL\_CURSORS\_2

The DS\_REPL\_CURSORS\_2 structure defines a set of [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) cursors for a given [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). This structure is a concrete representation of a sequence of [ReplUpToDateVector](#Section_8cb40d62a51d47e39b4e0837edffd61c) values; it is a superset of [DS\_REPL\_CURSORS](#Section_bfab2029039c442e8a924378d3a27473).

1. typedef struct {
2. DWORD cNumCursors;
3. DWORD dwEnumerationContext;
4. [size\_is(cNumCursors)] DS\_REPL\_CURSOR\_2 rgCursor[];
5. } DS\_REPL\_CURSORS\_2;

**cNumCursors:**  The count of items in the **rgCursor** array.

**dwEnumerationContext:**  The value a client uses to populate the **dwEnumerationContext** field of the request on a future call to [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47) to retrieve additional results. For an informative description of the sequencing issues associated with this field, see section [1.3.2](#Section_67c5a415a6c740988cf36ef8d173cfe8).

**rgCursor:**  A set of replication cursors.

##### DS\_REPL\_CURSOR\_2

The DS\_REPL\_CURSOR\_2 structure defines a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) cursor for a given [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). This structure is a concrete representation of a [ReplUpToDateVector](#Section_8cb40d62a51d47e39b4e0837edffd61c) value; it is a superset of [DS\_REPL\_CURSOR](#Section_cf960f2fc8fa4dfa9252f70164c14039).

1. typedef struct {
2. UUID uuidSourceDsaInvocationID;
3. USN usnAttributeFilter;
4. FILETIME ftimeLastSyncSuccess;
5. } DS\_REPL\_CURSOR\_2;

**uuidSourceDsaInvocationID:**  The [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**usnAttributeFilter:**  The [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) at which an [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) was applied on the DC.

**ftimeLastSyncSuccess:**  The time at which the last successful replication occurred from the DC identified by uuidDsa. Used for [**replication latency**](#gt_2352e9b3-ae08-4b5f-8858-bbca4ff4dd97) reporting only.

##### DS\_REPL\_CURSORS\_3W

The DS\_REPL\_CURSORS\_3W structure defines a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) cursor for a given [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). This structure is a concrete representation of a sequence of [ReplUpToDateVector](#Section_8cb40d62a51d47e39b4e0837edffd61c) values; it is a superset of [DS\_REPL\_CURSORS\_2](#Section_40366a5b9a48465fb7ac03f56334f76d).

1. typedef struct {
2. DWORD cNumCursors;
3. DWORD dwEnumerationContext;
4. [size\_is(cNumCursors)] DS\_REPL\_CURSOR\_3W rgCursor[];
5. } DS\_REPL\_CURSORS\_3W;

**cNumCursors:**  The count of items in the **rgCursor** array.

**dwEnumerationContext:**  The value a client uses to populate the **dwEnumerationContext** field of the request on a future call to [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47) to retrieve additional results. For an informative description of the sequencing issues associated with this field, see section [1.3.2](#Section_67c5a415a6c740988cf36ef8d173cfe8).

**rgCursor:**  A set of replication cursors.

##### DS\_REPL\_CURSOR\_3W

The DS\_REPL\_CURSOR\_3W structure defines a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) cursor for a given [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). This structure is a concrete representation of a [ReplUpToDateVector](#Section_8cb40d62a51d47e39b4e0837edffd61c) value; it is a superset of [DS\_REPL\_CURSOR\_2](#Section_40366a5b9a48465fb7ac03f56334f76d).

1. typedef struct {
2. UUID uuidSourceDsaInvocationID;
3. USN usnAttributeFilter;
4. FILETIME ftimeLastSyncSuccess;
5. [string] LPWSTR pszSourceDsaDN;
6. } DS\_REPL\_CURSOR\_3W;

**uuidSourceDsaInvocationID:**  The [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**usnAttributeFilter:**  The [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) at which an [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) was applied on the DC.

**ftimeLastSyncSuccess:**  The time at which the last successful replication occurred from the DC identified by uuidDsa. Used for [**replication latency**](#gt_2352e9b3-ae08-4b5f-8858-bbca4ff4dd97) reporting only.

**pszSourceDsaDN:**  The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) with an invocationId of **uuidSourceDsaInvocationID**.

##### DS\_REPL\_OBJ\_META\_DATA

The DS\_REPL\_OBJ\_META\_DATA structure defines a set of [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) [**stamps**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) for a given [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). This structure is a concrete representation of the sequence of [AttributeStamp](#Section_2973bb80c8ed450da98159639e09820b) values for all attributes of a given object.

1. typedef struct {
2. DWORD cNumEntries;
3. DWORD dwReserved;
4. [size\_is(cNumEntries)] DS\_REPL\_ATTR\_META\_DATA rgMetaData[];
5. } DS\_REPL\_OBJ\_META\_DATA;

**cNumEntries:**  The count of items in the **rgMetaData** array.

**dwReserved:**  Unused. MUST be 0 and ignored.

**rgMetaData:**  A set of attribute stamps.

##### DS\_REPL\_ATTR\_META\_DATA

The DS\_REPL\_ATTR\_META\_DATA structure defines an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) for a given [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). This structure is a concrete representation of an [AttributeStamp](#Section_2973bb80c8ed450da98159639e09820b).

1. typedef struct {
2. [string] LPWSTR pszAttributeName;
3. DWORD dwVersion;
4. FILETIME ftimeLastOriginatingChange;
5. UUID uuidLastOriginatingDsaInvocationID;
6. USN usnOriginatingChange;
7. USN usnLocalChange;
8. } DS\_REPL\_ATTR\_META\_DATA;

**pszAttributeName:**  The lDAPDisplayName of the attribute to which the stamp corresponds.

**dwVersion:**  The stamp version.

**ftimeLastOriginatingChange:**  The date and time at which the last [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) was made.

**uuidLastOriginatingDsaInvocationID:**  The [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that performed the last originating update.

**usnOriginatingChange:**  The [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) assigned to the last originating update by the DC that performed it.

**usnLocalChange:**  An implementation-specific value.

##### DS\_REPL\_OBJ\_META\_DATA\_2

The DS\_REPL\_OBJ\_META\_DATA\_2 structure defines a set of [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) [**stamps**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) for a given [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). This structure is a concrete representation of the sequence of [AttributeStamp](#Section_2973bb80c8ed450da98159639e09820b) values for all attributes of a given object; it is a superset of [DS\_REPL\_OBJ\_META\_DATA](#Section_a2ce73faaacd49d7911110ef08001963).

1. typedef struct {
2. DWORD cNumEntries;
3. DWORD dwReserved;
4. [size\_is(cNumEntries)] DS\_REPL\_ATTR\_META\_DATA\_2 rgMetaData[];
5. } DS\_REPL\_OBJ\_META\_DATA\_2;

**cNumEntries:**  The count of items in the **rgMetaData** array.

**dwReserved:**  Unused. MUST be 0 and ignored.

**rgMetaData:**  A set of attribute stamps.

##### DS\_REPL\_ATTR\_META\_DATA\_2

The DS\_REPL\_ATTR\_META\_DATA\_2 structure defines an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) for a given [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). This structure is a concrete representation of an [AttributeStamp](#Section_2973bb80c8ed450da98159639e09820b); it is a superset of [DS\_REPL\_ATTR\_META\_DATA](#Section_f7a10e539c454719a6414db15e385297).

1. typedef struct {
2. [string] LPWSTR pszAttributeName;
3. DWORD dwVersion;
4. FILETIME ftimeLastOriginatingChange;
5. UUID uuidLastOriginatingDsaInvocationID;
6. USN usnOriginatingChange;
7. USN usnLocalChange;
8. [string] LPWSTR pszLastOriginatingDsaDN;
9. } DS\_REPL\_ATTR\_META\_DATA\_2;

**pszAttributeName:**  The lDAPDisplayName of the attribute to which the stamp corresponds.

**dwVersion:**  The stamp version.

**ftimeLastOriginatingChange:**  The date and time at which the last [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) was made.

**uuidLastOriginatingDsaInvocationID:**  The [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that performed the last originating update.

**usnOriginatingChange:**  The [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) assigned to the last originating update by the DC that performed it.

**usnLocalChange:**  An implementation-specific value.

**pszLastOriginatingDsaDN:**  The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the nTDSDSA object with an invocationId of uuidLastOriginatingDsaInvocationID.

##### DS\_REPL\_KCC\_DSA\_FAILURESW

The DS\_REPL\_KCC\_DSA\_FAILURESW structure defines a set of [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that are in an error state with respect to [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb). This structure is a concrete representation of [KCCFailedConnections](#Section_eaffa80d8baf4784898ee9fbc7bd8296) and [KCCFailedLinks](#Section_fec285f37f034cfc89ac911f61c0c7d3).

1. typedef struct {
2. DWORD cNumEntries;
3. DWORD dwReserved;
4. [size\_is(cNumEntries)] DS\_REPL\_KCC\_DSA\_FAILUREW rgDsaFailure[];
5. } DS\_REPL\_KCC\_DSA\_FAILURESW;

**cNumEntries:**  The count of items in the **rgDsaFailure** array.

**dwReserved:**  Unused. MUST be 0 and ignored.

**rgDsaFailure:**  An array of [DS\_REPL\_KCC\_DSA\_FAILUREW](#Section_5d5ac3d8dc80401b9ca81e7a385024a4) structures.

##### DS\_REPL\_KCC\_DSA\_FAILUREW

The DS\_REPL\_KCC\_DSA\_FAILUREW structure defines a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that is in a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) error state. This structure is a concrete representation of a tuple in a [KCCFailedConnections](#Section_eaffa80d8baf4784898ee9fbc7bd8296) or [KCCFailedLinks](#Section_fec285f37f034cfc89ac911f61c0c7d3) sequence.

1. typedef struct {
2. [string] LPWSTR pszDsaDN;
3. UUID uuidDsaObjGuid;
4. FILETIME ftimeFirstFailure;
5. DWORD cNumFailures;
6. DWORD dwLastResult;
7. } DS\_REPL\_KCC\_DSA\_FAILUREW;

**pszDsaDN:**  The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the [**nTDSDSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8) corresponding to the DC.

**uuidDsaObjGuid:**  The [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the DC.

**ftimeFirstFailure:**  The date and time at which the DC entered an error state.

**cNumFailures:**  The number of errors that have occurred.

**dwLastResult:**  The [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b), as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.2, for the last error.

##### DS\_REPL\_PENDING\_OPSW

The DS\_REPL\_PENDING\_OPSW structure defines a sequence of [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) operations to be processed by a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). This structure is a concrete representation of [ReplicationQueue](#Section_6226aaa1178d45ff9e17815556739595).

1. typedef struct {
2. FILETIME ftimeCurrentOpStarted;
3. DWORD cNumPendingOps;
4. [size\_is(cNumPendingOps)] DS\_REPL\_OPW rgPendingOp[];
5. } DS\_REPL\_PENDING\_OPSW;

**ftimeCurrentOpStarted:**  The time when the current operation started.

**cNumPendingOps:**  The number of items in the **rgPendingOp** array.

**rgPendingOp:**  The sequence of replication operations to be performed.

##### DS\_REPL\_OPW

The DS\_REPL\_OPW structure defines a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) operation to be processed by a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). This structure is a concrete representation of a tuple in a [ReplicationQueue](#Section_6226aaa1178d45ff9e17815556739595) sequence.

1. typedef struct {
2. FILETIME ftimeEnqueued;
3. ULONG ulSerialNumber;
4. ULONG ulPriority;
5. DS\_REPL\_OP\_TYPE OpType;
6. ULONG ulOptions;
7. [string] LPWSTR pszNamingContext;
8. [string] LPWSTR pszDsaDN;
9. [string] LPWSTR pszDsaAddress;
10. UUID uuidNamingContextObjGuid;
11. UUID uuidDsaObjGuid;
12. } DS\_REPL\_OPW;

**ftimeEnqueued:**  The date and time at which the operation was requested.

**ulSerialNumber:**  The unique ID associated with the operation.

**ulPriority:**  A ULONG specifying the priority value of this operation. Tasks with a higher priority value are executed first. The priority is calculated by the server based on the type of operation and its parameters.

**OpType:**  An integer that indicates the type of operation, as defined in [DS\_REPL\_OP\_TYPE (section 5.46)](#Section_bf047cfe32bd43f693d3b67b05eaac66).

**ulOptions:**  The [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) flags.

**pszNamingContext:**  The [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the relevant [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

**pszDsaDN:**  The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the relevant DC's nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

**pszDsaAddress:**  The [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) of the relevant DC.

**uuidNamingContextObjGuid:**  The objectGUID of the NC root of the relevant NC replica.

**uuidDsaObjGuid:**  The [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the DC.

##### DS\_REPL\_ATTR\_VALUE\_META\_DATA

The DS\_REPL\_ATTR\_VALUE\_META\_DATA structure defines a sequence of [**link value stamps**](#gt_8878eaba-14b8-463b-94be-5784c1d24a85). This structure is a concrete representation of a sequence of [LinkValueStamp](#Section_6a9517897afa47dda96c83fc0e30aa3d) values.

1. typedef struct {
2. DWORD cNumEntries;
3. DWORD dwEnumerationContext;
4. [size\_is(cNumEntries)] DS\_REPL\_VALUE\_META\_DATA rgMetaData[];
5. } DS\_REPL\_ATTR\_VALUE\_META\_DATA;

**cNumEntries:**  The number of items in **rgMetaData** array.

**dwEnumerationContext:**  The value a client uses to populate the **dwEnumerationContext** field of the request on a future call to [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47) to retrieve additional results. For an informative description of the sequencing issues associated with this field, see section [1.3.2](#Section_67c5a415a6c740988cf36ef8d173cfe8).

**rgMetaData:**  The sequence of link value stamps.

##### DS\_REPL\_VALUE\_META\_DATA

The DS\_REPL\_VALUE\_META\_DATA structure defines a [**link value stamp**](#gt_8878eaba-14b8-463b-94be-5784c1d24a85). This structure is a concrete representation of a [LinkValueStamp](#Section_6a9517897afa47dda96c83fc0e30aa3d).

1. typedef struct {
2. [string] LPWSTR pszAttributeName;
3. [string] LPWSTR pszObjectDn;
4. DWORD cbData;
5. [size\_is(cbData), ptr] BYTE\* pbData;
6. FILETIME ftimeDeleted;
7. FILETIME ftimeCreated;
8. DWORD dwVersion;
9. FILETIME ftimeLastOriginatingChange;
10. UUID uuidLastOriginatingDsaInvocationID;
11. USN usnOriginatingChange;
12. USN usnLocalChange;
13. } DS\_REPL\_VALUE\_META\_DATA;

**pszAttributeName:**  The lDAPDisplayName of the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).

**pszObjectDn:**  The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

**cbData:**  The size, in bytes, of the **pbData** array.

**pbData:**  The binary\_value portion of the attribute value if the attribute is of syntax Object(DN-Binary), or the string\_value portion of the attribute value if the attribute is of syntax Object(DN-String); null otherwise.

**ftimeDeleted:**  The date and time at which the last [**replicated update**](#gt_2a923099-db0a-4932-af28-4354601e85c4) was made that deleted the value, or 0 if the value is not currently deleted.

**ftimeCreated:**  The date and time at which the first [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) was made.

**dwVersion:**  The [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) version.

**ftimeLastOriginatingChange:**  The date and time at which the last originating update was made.

**uuidLastOriginatingDsaInvocationID:**  The [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that performed the last originating update.

**usnOriginatingChange:**  The [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) assigned to the last originating update by the DC that performed the [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493).

**usnLocalChange:**  An implementation-specific value.

##### DS\_REPL\_ATTR\_VALUE\_META\_DATA\_2

The DS\_REPL\_ATTR\_VALUE\_META\_DATA\_2 structure defines a sequence of [**link value stamps**](#gt_8878eaba-14b8-463b-94be-5784c1d24a85). This structure is a concrete representation of a sequence of [LinkValueStamp](#Section_6a9517897afa47dda96c83fc0e30aa3d) values; it is a superset of [DS\_REPL\_ATTR\_VALUE\_META\_DATA](#Section_8e3dbb537f7f4c379095921fa9c0a4df).

1. typedef struct {
2. DWORD cNumEntries;
3. DWORD dwEnumerationContext;
4. [size\_is(cNumEntries)] DS\_REPL\_VALUE\_META\_DATA\_2 rgMetaData[];
5. } DS\_REPL\_ATTR\_VALUE\_META\_DATA\_2;

**cNumEntries:**  The number of items in the **rgMetaData** array.

**dwEnumerationContext:**  The value a client uses to populate the **dwEnumerationContext** field of the request on a future call to [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47) to retrieve additional results. For an informative description of the sequencing issues associated with this field, see section [1.3.2](#Section_67c5a415a6c740988cf36ef8d173cfe8).

**rgMetaData:**  The sequence of link value stamps.

##### DS\_REPL\_VALUE\_META\_DATA\_2

The DS\_REPL\_VALUE\_META\_DATA\_2 structure defines a [**link value stamp**](#gt_8878eaba-14b8-463b-94be-5784c1d24a85). This structure is a concrete representation of [LinkValueStamp](#Section_6a9517897afa47dda96c83fc0e30aa3d); it is a superset of [DS\_REPL\_VALUE\_META\_DATA](#Section_8e53006b9e1d48e6ba5fc675c0a98b3a).

1. typedef struct {
2. [string] LPWSTR pszAttributeName;
3. [string] LPWSTR pszObjectDn;
4. DWORD cbData;
5. [size\_is(cbData), ptr] BYTE\* pbData;
6. FILETIME ftimeDeleted;
7. FILETIME ftimeCreated;
8. DWORD dwVersion;
9. FILETIME ftimeLastOriginatingChange;
10. UUID uuidLastOriginatingDsaInvocationID;
11. USN usnOriginatingChange;
12. USN usnLocalChange;
13. [string] LPWSTR pszLastOriginatingDsaDN;
14. } DS\_REPL\_VALUE\_META\_DATA\_2;

**pszAttributeName:**  The lDAPDisplayName of the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).

**pszObjectDn:**  The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

**cbData:**  The size, in bytes, of the **pbData** array.

**pbData:**  The binary\_value portion of the attribute value if the attribute is of syntax Object(DN-Binary), or the string\_value portion of the attribute value if the attribute is of syntax Object(DN-String); null otherwise.

**ftimeDeleted:**  The date and time at which the last [**replicated update**](#gt_2a923099-db0a-4932-af28-4354601e85c4) was made that deleted the value, or 0 if the value is not currently deleted.

**ftimeCreated:**  The date and time at which the first [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) was made.

**dwVersion:**  The [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) version.

**ftimeLastOriginatingChange:**  The date and time at which the last originating update was made.

**uuidLastOriginatingDsaInvocationID:**  The [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that performed the last originating update.

**usnOriginatingChange:**  The [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) assigned to the last originating update by the DC that performed the [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493).

**usnLocalChange:**  An implementation-specific value.

**pszLastOriginatingDsaDN:**  The DN of the [**nTDSDSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8) with an invocationId of **uuidLastOriginatingDsaInvocationID**.

##### DS\_REPL\_CLIENT\_CONTEXTS

The DS\_REPL\_CLIENT\_CONTEXTS structure defines a set of active [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) client connections. This structure is a concrete representation of [RPCClientContexts](#Section_65d838f52f694c228b263340182dcde1).

1. typedef struct {
2. [range(0,10000)] DWORD cNumContexts;
3. DWORD dwReserved;
4. [size\_is(cNumContexts)] DS\_REPL\_CLIENT\_CONTEXT rgContext[];
5. } DS\_REPL\_CLIENT\_CONTEXTS;

**cNumContexts:**  The number of items in the **rgContext** array.

**dwReserved:**  Unused. MUST be 0 and ignored.

**rgContext:**  A set of active RPC client connections.

##### DS\_REPL\_CLIENT\_CONTEXT

The DS\_REPL\_CLIENT\_CONTEXT structure defines an active [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) client connection. This structure is a concrete representation of a tuple in an [RPCClientContexts](#Section_65d838f52f694c228b263340182dcde1) sequence.

1. typedef struct {
2. ULONGLONG hCtx;
3. LONG lReferenceCount;
4. BOOL fIsBound;
5. UUID uuidClient;
6. DSTIME timeLastUsed;
7. ULONG IPAddr;
8. int pid;
9. } DS\_REPL\_CLIENT\_CONTEXT;

**hCtx:**  The unique ID of the client context.

**lReferenceCount:**  The number of references to the context.

**fIsBound:**  True if and only if the context has not yet been closed by the [IDL\_DRSUnbind](#Section_49eb17c9b6a94ceabef866abda8a7850) method.

**uuidClient:**  Zeros, or the value pointed to by the *puuidClientDsa* parameter to [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d).

**timeLastUsed:**  The date and time at which this context was last used in an RPC method call.

**IPAddr:**  The IPv4 address of the client. If the client is connected with IPv6, this field MUST be 0.

**pid:**  The process ID specified by the client in the *pextClient* parameter to IDL\_DRSBind.

##### DS\_REPL\_SERVER\_OUTGOING\_CALLS

The DS\_REPL\_SERVER\_OUTGOING\_CALLS structure defines a set of outstanding requests from this [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) to other DCs. This structure is a concrete representation of [RPCOutgoingContexts](#Section_b9f465938a4041869ecebc2612b4c3f4).

1. typedef struct {
2. [range(0,256)] DWORD cNumCalls;
3. DWORD dwReserved;
4. [size\_is(cNumCalls)] DS\_REPL\_SERVER\_OUTGOING\_CALL rgCall[];
5. } DS\_REPL\_SERVER\_OUTGOING\_CALLS;

**cNumCalls:**  The number of items in the **rgCall** array.

**dwReserved:**  Unused. MUST be 0 and ignored.

**rgCall:**  A set of outstanding requests from this DC to other DCs.

##### DS\_REPL\_SERVER\_OUTGOING\_CALL

The DS\_REPL\_SERVER\_OUTGOING\_CALL structure defines an outstanding request from this [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) to another DC. This structure is a concrete representation of a tuple from an [RPCOutgoingContexts](#Section_b9f465938a4041869ecebc2612b4c3f4) sequence.

1. typedef struct {
2. [string] LPWSTR pszServerName;
3. BOOL fIsHandleBound;
4. BOOL fIsHandleFromCache;
5. BOOL fIsHandleInCache;
6. DWORD dwThreadId;
7. DWORD dwBindingTimeoutMins;
8. DSTIME dstimeCreated;
9. DWORD dwCallType;
10. } DS\_REPL\_SERVER\_OUTGOING\_CALL;

**pszServerName:**  The [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) of the server.

**fIsHandleBound:**  True if and only if the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method has completed and the [IDL\_DRSUnbind](#Section_49eb17c9b6a94ceabef866abda8a7850) method has not yet been called.

**fIsHandleFromCache:**  True if and only if the context handle used was retrieved from the cache.

**fIsHandleInCache:**  True if and only if the context handle is still in the cache.

**dwThreadId:**  The thread ID of the thread that is using the context.

**dwBindingTimeoutMins:**  If the context is set to be canceled, the time-out in minutes.

**dstimeCreated:**  The date and time when the context was created.

**dwCallType:**  The call that the client is waiting on. MUST be one of the values in the following table.

| Value | Meaning |
| --- | --- |
| 2 | IDL\_DRSBind |
| 3 | IDL\_DRSUnbind |
| 4 | [IDL\_DRSReplicaSync](#Section_25c71d91051f4c26977fa70892f29b00) |
| 5 | [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) |
| 6 | [IDL\_DRSUpdateRefs](#Section_a273bbcfaeca46088ad4127d3e597cd4) |
| 7 | [IDL\_DRSReplicaAdd](#Section_7219df914eea494f88e3780d40d2d559) |
| 8 | [IDL\_DRSReplicaDel](#Section_1420a9bf9267464da6d57676472d7f1d) |
| 9 | [IDL\_DRSVerifyNames](#Section_80739a29e8ed44788490475a18e9779d) |
| 10 | [IDL\_DRSGetMemberships](#Section_d5ace4527cdd4d50bb6439b4c55180a2) |
| 11 | [IDL\_DRSInterDomainMove](#Section_595b2fef493b4b1db60de7a1a3345b0e) |
| 12 | [IDL\_DRSGetNT4ChangeLog](#Section_6e000eb660fd4d6cae82bb6479df02fa) |
| 13 | [IDL\_DRSCrackNames](#Section_9b4bfb4466564404bcc8dc88111658b3) |
| 14 | [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e) |
| 15 | [IDL\_DRSGetMemberships2](#Section_d4e67cc32ee14b2b8055cebefc556252) |
| 16 | [IDL\_DRSGetObjectExistence](#Section_6355d4f5f5564527adde37afba2fcf56) |
| 17 | [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47) |
| 18 | [IDL\_DRSWriteSPN](#Section_8b129dc8ed4545379555b6fef764ab7d) |

#### Method-Specific Abstract Types and Procedures

##### GetDNFromInvocationID

1. procedure GetDNFromInvocationID(invocationID: GUID): DN

Returns the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that has the specified [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a). If there is no such nTDSDSA object, the results are unconstrained and the resulting behavior of protocol elements that use this returned DN are also unconstrained.

##### GetDNFromObjectGuid

1. procedure GetDNFromObjectGuid(guid: GUID): DN

Returns the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) with the specified object [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1). This is represented by the following expression.

1. obj := select one o from all where (o!objectGUID = guid)
2. return obj.dn

##### GetNCs

1. procedure GetNCs(): set of DSName

Returns a set containing the [DSName](#Section_a0d5477a522946b9890a54b924d487d1)s of all [**NCs**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) hosted by this server.

##### GetUpToDatenessVector

1. procedure GetUpToDatenessVector(nc: DSName): sequence of ReplUpToDateVector

Returns a sequence of [ReplUpToDateVector (section 5.166)](#Section_8cb40d62a51d47e39b4e0837edffd61c), sorted in ascending order by the **uuidDsa** field. The entries are retrieved from *nc*!replUpToDateVector plus an additional entry with **uuidDsa** set to the [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of this server, **usnHighPropUpdate** set to rootDSE!highestCommittedUSN, and **timeLastSyncSuccess** set to the current time.

#### Server Behavior of the IDL\_DRSGetReplInfo Method

*Informative summary of behavior*: This method retrieves the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state information of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). Based on the value of the **InfoType** field in the request message, different information is returned, which is summarized in the definition of DS\_REPL\_INFO in section [4.1.13.1.4](#Section_d5afb4d307e2451b87d072f4fb2c2eaf).

1. ULONG
2. IDL\_DRSGetReplInfo(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_GETREPLINFO\_REQ \*pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DRS\_MSG\_GETREPLINFO\_REPLY \*pmsgOut)
10. msgIn: DRS\_MSG\_GETREPLINFO\_REQ\_V2
11. infoType: DWORD
12. fAccessGranted: boolean
13. infoTypeValid: boolean
14. defaultNC: DSName
15. object: DSName
16. enumerationContext: DWORD
17. baseIndex: DWORD
18. endIndex: DWORD
19. ncs: set of DSName
20. nc: DSName
21. i, j: DWORD
22. r: RepsFrom
23. q: RepsTo
24. pNeighbor: ADDRESS OF DS\_REPL\_NEIGHBORW
25. utd: sequence of ReplUpToDateVector
26. pCursor: ADDRESS OF DS\_REPL\_CURSOR
27. pCursor2: ADDRESS OF DS\_REPL\_CURSOR\_2
28. pCursor3: ADDRESS OF DS\_REPL\_CURSOR\_3W
29. a: ATTRTYP
30. attr: ATTRTYP
31. attrs: set of ATTRTYP
32. attrsSeq: sequence of ATTRTYP
33. s: AttributeStamp
34. stamp: LinkValueStamp
35. pObjMetaData: ADDRESS OF DS\_REPL\_OBJ\_META\_DATA
36. pObjMetaData2: ADDRESS OF DS\_REPL\_OBJ\_META\_DATA\_2
37. values: set of attribute value
38. valuesSeq: sequence of attribute value
39. ls: LinkValueStamp
40. pAttrValueMetaData: ADDRESS OF DS\_REPL\_ATTR\_VALUE\_META\_DATA
41. pAttrValueMetaData2: ADDRESS OF DS\_REPL\_ATTR\_VALUE\_META\_DATA\_2
42. pFailedConnection: ADDRESS OF DS\_REPL\_KCC\_DSA\_FAILUREW
43. pFailedLink: ADDRESS OF DS\_REPL\_KCC\_DSA\_FAILUREW
44. pPendingOp: ADDRESS OF DS\_REPL\_OPW
45. pClientContext: ADDRESS OF DS\_REPL\_CLIENT\_CONTEXT
46. pOutgoingContext: ADDRESS OF DS\_REPL\_SERVER\_OUTGOING\_CALL
47. v: attribute value
48. ValidateDRSInput(hDrs, 19)
49. if dwInVersion = 1 then
50. infoType = pmsgIn^ V1.InfoType
51. else
52. infoType = pmsgIn^ V2.InfoType
53. endif
54. pdwOutVersion^ := infoType
55. if infoType = DS\_REPL\_INFO\_NEIGHBORS then
56. pmsgOut^.pNeighbors := null
57. else if infoType = DS\_REPL\_INFO\_CURSORS\_FOR\_NC then
58. pmsgOut^.pCursors := null
59. else if infoType = DS\_REPL\_INFO\_METADATA\_FOR\_OBJ then
60. pmsgOut^.pObjMetaData := null
61. else if infoType = DS\_REPL\_INFO\_KCC\_DSA\_CONNECT\_FAILURES then
62. pmsgOut^.pConnectFailures := null
63. else if infoType = DS\_REPL\_INFO\_KCC\_DSA\_LINK\_FAILURES then
64. pmsgOut^.pLinkFailures := null
65. else if infoType = DS\_REPL\_INFO\_PENDING\_OPS then
66. pmsgOut^.pPendingOps := null
67. else if infoType = DS\_REPL\_INFO\_METADATA\_FOR\_ATTR\_VALUE then
68. pmsgOut^.pAttrValueMetaData := null
69. else if infoType = DS\_REPL\_INFO\_CURSORS\_2\_FOR\_NC then
70. pmsgOut^.pCursors2 := null
71. else if infoType = DS\_REPL\_INFO\_CURSORS\_3\_FOR\_NC then
72. pmsgOut^.pCursors3 := null
73. else if infoType = DS\_REPL\_INFO\_METADATA\_2\_FOR\_OBJ then
74. pmsgOut^.pObjMetaData2 := null
75. else if infoType = DS\_REPL\_INFO\_METADATA\_2\_FOR\_ATTR\_VALUE then
76. pmsgOut^.pAttrValueMetaData2 := null
77. else if infoType = DS\_REPL\_INFO\_SERVER\_OUTGOING\_CALLS then
78. pmsgOut^.pServerOutgoingCalls := null
79. else if infoType = DS\_REPL\_INFO\_UPTODATE\_VECTOR\_V1 then
80. pmsgOut^.pUpToDateVec := null
81. else if infoType = DS\_REPL\_INFO\_CLIENT\_CONTEXTS then
82. pmsgOut^.pClientContexts := null
83. else if infoType = DS\_REPL\_INFO\_REPSTO then
84. pmsgOut^.pRepsTo := null
85. endif
86. /\* Validate the version of the request message \*/
87. if (dwInVersion ≠ 1 and dwInVersion ≠ 2) then
88. return ERROR\_REVISION\_MISMATCH
89. endif
90. if dwInVersion = 1 then
91. msgIn := pmsgIn^.V1
92. else
93. msgIn := pmsgIn^.V2
94. endif
95. /\* For some of the request types, paging is supported. For these
96. \* cases, a starting index into the result set is needed based on
97. \* what has already been returned in a previous call. Only version 2
98. \* request messages provide a mechanism for the client to supply the
99. \* context information from a previous call. \*/
100. if dwInVersion = 1 then
101. baseIndex := 0
102. else
103. if msgIn.dwEnumerationContext = 0xffffffff then
104. /\* No more data is available. \*/
105. return ERROR\_NO\_MORE\_ITEMS
106. endif
107. baseIndex := msgIn.dwEnumerationContext
108. endif
109. /\* Perform the necessary access checks. \*/
110. defaultNC := DefaultNC()
111. fAccessGranted := false
112. infoTypeValid := false
113. object := msgIn.pszObjectDN
114. if (infoType = DS\_REPL\_INFO\_NEIGHBORS and object ≠ null) then
115. infoTypeValid := true
116. fAccessGranted :=
117. AccessCheckAttr(object, repsFrom, RIGHT\_DS\_READ\_PROPERTY) or
118. AccessCheckCAR(object, DS-Replication-Manage-Topology) or
119. AccessCheckCAR(object, DS-Replication-Monitor-Topology)
120. endif
121. if (infoType = DS\_REPL\_INFO\_NEIGHBORS and object = null) then
122. infoTypeValid := true
123. fAccessGranted :=
124. AccessCheckAttr(defaultNC, repsFrom, RIGHT\_DS\_READ\_PROPERTY) or
125. AccessCheckCAR(defaultNC, DS-Replication-Manage-Topology) or
126. AccessCheckCAR(defaultNC, DS-Replication-Monitor-Topology)
127. endif
128. if (infoType = DS\_REPL\_INFO\_REPSTO and object ≠ null) then
129. infoTypeValid := true
130. fAccessGranted :=
131. AccessCheckAttr(object, repsTo, RIGHT\_DS\_READ\_PROPERTY) or
132. AccessCheckCAR(object, DS-Replication-Manage-Topology) or
133. AccessCheckCAR(object, DS-Replication-Monitor-Topology)
134. endif
135. if (infoType = DS\_REPL\_INFO\_REPSTO and object = null) then
136. infoTypeValid := true
137. fAccessGranted :=
138. AccessCheckAttr(defaultNC, repsTo, RIGHT\_DS\_READ\_PROPERTY) or
139. AccessCheckCAR(defaultNC, DS-Replication-Manage-Topology) or
140. AccessCheckCAR(defaultNC, DS-Replication-Monitor-Topology)
141. endif
142. if (infoType in {DS\_REPL\_INFO\_CURSORS\_FOR\_NC,
143. DS\_REPL\_INFO\_CURSORS\_2\_FOR\_NC,
144. DS\_REPL\_INFO\_CURSORS\_3\_FOR\_NC,
145. DS\_REPL\_INFO\_UPTODATE\_VECTOR\_V1} and
146. object ≠ null) then
147. infoTypeValid := true
148. fAccessGranted :=
149. AccessCheckAttr(
150. object, replUpToDateVector, RIGHT\_DS\_READ\_PROPERTY) or
151. AccessCheckCAR(object, DS-Replication-Manage-Topology) or
152. AccessCheckCAR(object, DS-Replication-Monitor-Topology)
153. endif
154. if infoType in {DS\_REPL\_INFO\_METADATA\_FOR\_OBJ,
155. DS\_REPL\_INFO\_METADATA\_2\_FOR\_OBJ,
156. DS\_REPL\_INFO\_METADATA\_FOR\_ATTR\_VALUE,
157. DS\_REPL\_INFO\_METATDATA\_2\_FOR\_ATTR\_VALUE} then
158. if object = null then
159. return ERROR\_INVALID\_PARAMETER
160. endif
161. if not ObjExists(object) then
162. if object.dn = null then
163. return ERROR\_DS\_DRA\_BAD\_DN
164. else
165. return ERROR\_DS\_OBJ\_NOT\_FOUND
166. endif
167. endif
168. infoTypeValid := true
169. fAccessGranted :=
170. AccessCheckAttr(object,
171. replPropertyMetaData,
172. RIGHT\_DS\_READ\_PROPERTY) or
173. AccessCheckCAR(object, DS-Replication-Manage-Topology) or
174. AccessCheckCAR(object, DS-Replication-Monitor-Topology)
175. endif
176. if infoType in {DS\_REPL\_INFO\_PENDING\_OPS,
177. DS\_REPL\_INFO\_KCC\_DSA\_CONNECT\_FAILURES,
178. DS\_REPL\_INFO\_KCC\_DSA\_LINK\_FAILURES,
179. DS\_REPL\_INFO\_CLIENT\_CONTEXTS,
180. DS\_REPL\_INFO\_SERVER\_OUTGOING\_CALLS} then
181. infoTypeValid := true
182. fAccessGranted :=
183. AccessCheckCAR(defaultNC, DS-Replication-Manage-Topology) or
184. AccessCheckCAR(defaultNC, DS-Replication-Monitor-Topology)
185. endif
186. if not infoTypeValid then
187. return ERROR\_INVALID\_PARAMETER
188. endif
189. if not fAccessGranted then
190. return ERROR\_DS\_DRA\_ACCESS\_DENIED
191. endif
192. /\* Based on the type of information requested, the corresponding
193. \* information is retrieved and the response message constructed \*/
194. /\* DS\_REPL\_INFO\_NEIGHBORS/DS\_REPL\_INFO\_REPSTO \*/
195. if infoType in {DS\_REPL\_INFO\_NEIGHBORS, DS\_REPL\_INFO\_REPSTO}
196. /\* If an object is specified, it must be an NC root. \*/
197. nc := object
198. if nc ≠ null then
199. ncs := {nc}
200. else
201. ncs := GetNCs()
202. endif
204. if infoType = DS\_REPL\_INFO\_NEIGHBORS then
205. i := 0
206. j := 0
207. foreach nc in ncs
208. foreach r in nc!repsFrom
209. /\* The ordering of ncs hosted by the server and the values of
210. \* repsFrom for each nc is arbitrary but consistent from call
211. \* to call on a server. \*/
212. /\* If a source server GUID is specified, only information for
213. \* that server is returned. \*/
214. If (msgIn.uuidSourceDsaGuid = NULLGUID or
215. msgIn.uuidSourceDsaGuid = r.uuidDsa) then
216. if i >= baseIndex then
217. pNeighbor := ADR(pmsgOut^.pNeighbors^.rgNeighbor[j])
218. pNeighbor^.pszSourceDsaAddress := r.naDsa
219. pNeighbor^.uuidSourceDsaObjGuid := r.uuidDsa
220. pNeighbor^.pszSourceDsaDN :=
221. GetDNFromObjectGuid(r.uuidDsa)
222. pNeighbor^.pszNamingContext := nc!distinguishedName
223. /\* If a naming context is specified in the request,
224. \* the uuidNamingContextObjGuid field of the response
225. \* is set to the NULL GUID. \*/
226. if object ≠ null then
227. pNeighbor^.uuidNamingContextObjGuid := NULLGUID
228. else
229. pNeighbor^.uuidNamingContextObjGuid := nc!objectGUID
230. endif
231. pNeighbor^.pszAsyncIntersiteTransportDN :=
232. GetDNFromObjectGuid(r.uuidTransportObj)
233. pNeighbor^.uuidSourceDsaInvocationID := r.uuidInvocId
234. pNeighbor^.uuidAsyncIntersiteTransportObjGuid :=
235. r.uuidTransportObj
236. pNeighbor^.usnLastObjChangeSynced :=
237. r.usnVec.usnHighObjUpdate
238. pNeighbor^.usnAttributeFilter :=
239. r.usnVec.usnHighPropUpdate
240. pNeighbor^.ftimeLastSyncSuccess := r.timeLastSuccess
241. pNeighbor^.ftimeLastSyncAttempt := r.timeLastAttempt
242. pNeighbor^.dwLastSyncResult := r.ulResultLastAttempt
243. pNeighbor^.cNumConsecutiveSyncFailures :=
244. r.cConsecutiveFailures
245. /\* Only a subset of the possible DRS\_OPTIONS in r.options
246. \* are preserved in pNeighbor^.dwReplicaFlags.
247. \* See section 5.169 repsFrom, RepsFrom for more info. \*/
248. pNeighbor^.dwReplicaFlags := {}
249. foreach flag in { DRS\_WRIT\_REP,
250. DRS\_INIT\_SYNC,
251. DRS\_PER\_SYNC,
252. DRS\_MAIL\_REP,
253. DRS\_DISABLE\_AUTO\_SYNC,
254. DRS\_DISABLE\_PERIODIC\_SYNC,
255. DRS\_USE\_COMPRESSION,
256. DRS\_TWOWAY\_SYNC,
257. DRS\_NONGC\_RO\_REP,
258. DRS\_FULL\_SYNC\_IN\_PROGRESS,
259. DRS\_FULL\_SYNC\_PACKET,
260. DRS\_REF\_GCSPN,
261. DRS\_NEVER\_SYNCED,
262. DRS\_SPECIAL\_SECRET\_PROCESSING,
263. DRS\_PREEMPTED,
264. DRS\_NEVER\_NOTIFY,
265. DRS\_SYNC\_PAS}
266. if flag in r.options then
267. pNeighbor^.dwReplicaFlags := pNeighbor^.dwReplicaFlags + flag
268. endif
269. endfor
270. j := j + 1
271. endif
272. i := i + 1
273. endif
274. endfor
275. endfor
276. pmsgOut^.pNeighbors^.cNumNeighbors := j
277. else
278. /\* DS\_REPL\_INFO\_REPSTO case. \*/
279. i := 0
280. j := 0
281. foreach nc in ncs
282. foreach q in nc!repsTo
283. /\* The ordering of ncs hosted by the server and the values of
284. \* repsTo for each nc is arbitrary but consistent from call
285. \* to call on a server. \*/
286. if i >= baseIndex then
287. pNeighbor := ADR(pmsgOut^.pRepsTo^.rgNeighbor[j])
288. pNeighbor^.pszSourceDsaAddress := q.naDsa
289. pNeighbor^.ftimeLastSyncSuccess := q.timeLastSuccess
290. pNeighbor^.ftimeLastSyncAttempt := q.timeLastAttempt
291. pNeighbor^.dwLastSyncResult := q.ulResultLastAttempt
292. pNeighbor^.cNumConsecutiveSyncFailures :=
293. q.cConsecutiveFailures
294. pNeighbor^.uuidSourceDsaObjGuid := q.uuidDsa
295. pNeighbor^.pszSourceDsaDN := GetDNFromObjectGuid(q.uuidDsa)
296. pNeighbor^.pszNamingContext := nc!distinguishedName
297. /\* If a naming context is specified in the request,
298. \* the uuidNamingContextObjGuid field of the response
299. \* is set to the NULL GUID. \*/
300. if object ≠ null then
301. pNeighbor^.uuidNamingContextObjGuid := NULLGUID
302. else
303. pNeighbor^.uuidNamingContextObjGuid := nc!objectGUID
304. endif
305. /\* Only a subset of the possible DRS\_OPTIONS in q.options
306. \* are preserved in pNeighbor^.dwReplicaFlags.
307. \* See section 5.170 repsTo, RepsTo for more info. \*/
308. pNeighbor^.dwReplicaFlags := {}
309. foreach flag in { DRS\_WRIT\_REP,
310. DRS\_INIT\_SYNC,
311. DRS\_PER\_SYNC,
312. DRS\_MAIL\_REP,
313. DRS\_DISABLE\_AUTO\_SYNC,
314. DRS\_DISABLE\_PERIODIC\_SYNC,
315. DRS\_USE\_COMPRESSION,
316. DRS\_TWOWAY\_SYNC,
317. DRS\_NONGC\_RO\_REP,
318. DRS\_FULL\_SYNC\_IN\_PROGRESS,
319. DRS\_FULL\_SYNC\_PACKET,
320. DRS\_REF\_GCSPN,
321. DRS\_NEVER\_SYNCED,
322. DRS\_SPECIAL\_SECRET\_PROCESSING,
323. DRS\_PREEMPTED,
324. DRS\_NEVER\_NOTIFY,
325. DRS\_SYNC\_PAS}
326. if flag in q.options then
327. pNeighbor^.dwReplicaFlags := pNeighbor^.dwReplicaFlags + flag
328. endif
329. endfor
330. j := j + 1
331. endif
332. i := i + 1
333. endfor
334. endfor
335. pmsgOut^.pRepsTo^.cNumNeighbors := j
336. endif
337. endif
338. /\* DS\_REPL\_INFO\_METADATA\_FOR\_OBJ/DS\_REPL\_INFO\_METADATA\_2\_FOR\_OBJ \*/
339. if infoType in {DS\_REPL\_INFO\_METADATA\_FOR\_OBJ,
340. DS\_REPL\_INFO\_METADATA\_2\_FOR\_OBJ) then
341. /\* Enumerate all the replicated attributes \*/
342. attrsSeq := ReplicatedAttributes()
343. i := 0
344. j := 0
345. while (i < attrsSeq.length)
346. attr := attrsSeq[i]
347. s := AttrStamp(object, attr)
348. if (IsForwardLinkAttribute(attr) and
349. dwInVersion = 2 and
350. DS\_REPL\_INFO\_FLAG\_IMPROVE\_LINKED\_ATTRS in msgIn.ulFlags)
351. then
352. ls := null
353. foreach v in GetAttrVals(object, attr, true)
354. stamp := LinkStamp(object, attr, v)
355. /\* If v was last updated in win2k forest mode
356. \* then it does not have LinkValueStamp associated with it.
357. \* LinkStamp() returns null in that case. \*/
358. if stamp ≠ null and LinkValueStampCompare(stamp, ls) > 0 then
359. ls := stamp;
360. endif
361. endfor
362. if s = null then
363. s := 0 /\* An AttributeStamp with 0 for all fields. \*/
364. endif
365. /\* Improve the stamp with the link value stamp. \*/
366. s.dwVersion := ls.dwVersion
367. s.timeChanged := ls.timeChanged
368. s.uuidOriginating := NULLGUID
369. s.usnOriginating := ls.usnOriginating
370. endif
371. if s ≠ null then
372. if i >= baseIndex
373. if infoType = DS\_REPL\_INFO\_METADATA\_FOR\_OBJ then
374. pObjMetaData := ADR(pmsgOut^.pObjMetaData^.rgMetaData[j])
375. pObjMetaData^.pszAttributeName := attr
376. pObjMetaData^.dwVersion := s.dwVersion
377. pObjMetaData^.ftimeLastOriginatingChange := s.timeChanged
378. pObjMetaData^.uuidLastOriginatingDsaInvocationID :=
379. s.uuidOriginating
380. pObjMetaData^.usnOriginatingChange := s.usnOriginating
381. pObjMetaData^.usnLocalChange :=
382. An implementation-specific value that the server
383. maintains for replicated attributes
384. else
385. pObjMetaData2 := ADR(pmsgOut^.pObjMetaData2^.rgMetaData[j])
386. pObjMetaData2^.pszAttributeName := attr
387. pObjMetaData2^.dwVersion := s.dwVersion
388. pObjMetaData2^.ftimeLastOriginatingChange := s.timeChanged
389. pObjMetaData2^.uuidLastOriginatingDsaInvocationID :=
390. s.uuidOriginating
391. pObjMetaData2^.usnOriginatingChange := s.usnOriginating
392. pObjMetaData2^.usnLocalChange :=
393. An implementation-specific value that the server
394. maintains for replicated attributes
395. pObjMetaData2^.pszLastOriginatingDsaDN :=
396. GetDNFromInvocationID(s.uuidOriginating)
397. endif
398. j := j + 1
399. endif
400. i := i + 1
401. endif
402. endwhile
403. if infoType = DS\_REPL\_INFO\_METADATA\_FOR\_OBJ then
404. pmsgOut^.pObjMetaData^.cNumEntries = j
405. else
406. pmsgOut^.pObjMetaData2^.cNumEntries = j
407. endif
408. endif
409. /\* DS\_REPL\_INFO\_CURSORS\_FOR\_NC \*/
410. if infoType = DS\_REPL\_INFO\_CURSORS\_FOR\_NC then
411. /\* The NC root object must be specified \*/
412. nc := object
413. /\* Parameter validation \*/
414. if nc = null then
415. return ERROR\_INVALID\_PARAMETER
416. endif
417. if not FullReplicaExists(nc) and
418. not PartialGCReplicaExists(nc) then
419. return ERROR\_DS\_DRA\_BAD\_NC
420. endif
421. utd := GetUpToDatenessVector(nc)
422. i := baseIndex
423. j := 0
424. while i < utd.length
425. pCursor := ADR(pmsgOut^.pCursors^.rgCursor[j])
426. pCursor^.uuidSourceDsaInvocationID := utd[i].uuidDsa
427. pCursor^.usnAttributeFilter := utd[i].usnHighPropUpdate
428. i := i + 1
429. j := j + 1
430. endwhile
431. pmsgOut^.pCursors^.cNumCursors := j
432. endif
433. /\* DS\_REPL\_INFO\_CURSORS\_2\_FOR\_NC/ DS\_REPL\_INFO\_CURSORS\_3\_FOR\_NC \*/
434. if infoType in {DS\_REPL\_INFO\_CURSORS\_2\_FOR\_NC,
435. DS\_REPL\_INFO\_CURSORS\_3\_FOR\_NC} then
436. /\* The NC root object must be specified. \*/
437. nc := object
438. /\* Parameter validation. \*/
439. if (nc = null) then
440. return ERROR\_INVALID\_PARAMETER
441. endif
442. if not FullReplicaExists(nc) and
443. not PartialGCReplicaExists(nc) then
444. return ERROR\_DS\_DRA\_BAD\_NC
445. endif
446. i := baseIndex
447. j := 0
448. utd := GetUpToDatenessVector(nc)
449. /\* A maximum of 1000 items will be sent in each call. \*/
450. if utd.length - baseIndex - 1 > 1000 then
451. endIndex = baseIndex + 1000
452. else
453. endIndex = utd.length
454. endif
455. while i < endIndex
456. if infoType = DS\_REPL\_INFO\_CURSORS\_2\_FOR\_NC then
457. pCursor2 := ADR(pmsgOut^.pCursors2^.rgCursor[j])
458. pCursor2^.uuidSourceDsaInvocationID := utd[i].uuidDsa
459. pCursor2^.usnAttributeFilter := utd[i].usnHighPropUpdate
460. pCursor2^.ftimeLastSyncSucess := utd[i].timeLastSyncSuccess
461. else
462. pCursor3 := ADR(pmsgOut^.pCursor3^.rgCursor[j])
463. pCursor3^.uuidSourceDsaInvocationID := utd[i].uuidDsa
464. pCursor3^.usnAttributeFilter := utd[i].usnHighPropUpdate
465. pCursor3^.ftimeLastSyncSucess := utd[i].timeLastSyncSuccess
466. pCursor3^.pszSourceDsaDN :=
467. GetDNFromInvocationID(utd[i].uuidDsa)
468. endif
469. j := j + 1
470. i := i + 1
471. endwhile
472. if infoType = DS\_REPL\_INFO\_CURSORS\_2\_NC then
473. pmsgOut^.pCursors2^.cNumCursors := j
474. else
475. pmsgOut^.pCursors3^.cNumCursors := j
476. endif
477. if i < utd.length - 1 then
478. /\* Not all items could be sent back in this call, so save the
479. \* index of the first item to be sent in the next call. \*/
480. If infoType = DS\_REPL\_INFO\_CURSORS\_2\_NC then
481. pmsgOut^.pCursor2^.dwEnumerationContext := i
482. else
483. pmsgOut^.pCursors3^.dwEnumerationContext := i
484. endif
485. else
486. /\* No more data is available. \*/
487. If infoType = DS\_REPL\_INFO\_CURSORS\_2\_NC then
488. pmsgOut^.pCursor2^.dwEnumerationContext := 0xffffffff
489. else
490. pmsgOut^.pCursors3^.dwEnumerationContext := 0xffffffff
491. endif
492. endif
493. endif
495. /\* DS\_REPL\_INFO\_UPTODATE\_VECTOR\_V1 \*/
496. if infoType = DS\_REPL\_INFO\_UPTODATE\_VECTOR\_V1 then
497. /\* The NC root object must be specified. \*/
498. nc := object
499. /\* Parameter validation. \*/
500. if (nc = null) then
501. return ERROR\_INVALID\_PARAMETER
502. endif
503. utd := GetUpToDatenessVector(nc)
504. for i := 0 to utd.length - 1
505. pCursor := ADR(pmsgOut^.pUpToDateVec^.rgCursors[i])
506. pCursor^.uuidSourceDsaInvocationID := utd[i].uuidDsa
507. pCursor^.usnAttributeFilter := utd[i].usnHighPropUpdate
508. endfor
509. pmsgOut^.pUpToDateVec^.cNumCursors := utd.length
510. endif
511. /\* DS\_REPL\_INFO\_METADATA\_FOR\_ATTR\_VALUE/
512. \* DS\_REPL\_INFO\_METADATA\_2\_FOR\_ATTR\_VALUE \*/
513. if infoType in {DS\_REPL\_INFO\_METADATA\_FOR\_ATTR\_VALUE,
514. DS\_REPL\_INFO\_METADATA\_2\_FOR\_ATTR\_VALUE} then
515. /\* If the attribute name is specified it must be a link
516. \* attribute. \*/
517. attrs := select all a in Link Attributes of object
518. if (pmsgIn^.V2.pszAttributeNameValue ≠ null and
519. pmsgIn^.V2.pszAttributeNameValue not in attrs) then
520. return ERROR\_DS\_WRONG\_LINKED\_ATT\_SYNTAX
521. endif
522. /\* If the attribute name is not specified, replication state for a
523. \* link attribute of the object which has a value is returned. \*/
524. if (pmsgIn^.V2.pszAttributeNameValue ≠ null) then
525. attr := pmsgIn^.V2.pszAttributeNameValue
526. else
527. attrsSeq := select all a in attrs where
528. GetAttrVals(object, a, true) ≠ null
529. attr := attrsSeq[0]
530. endif
531. if attr ≠ null then
532. valuesSeq := GetAttrVals(object, attr, true)
533. /\* If a start value has been specified, then start at the first
534. \* occurrence of that value in the sequence of values, otherwise
535. \* start at the index determined from the enumeration context
536. \* which specifies the index of the next value to be returned. \*/
537. if (pmsgIn^.V2.pszValueDN ≠ null and
538. Syntax(attr) = Object(DS-DN)) then
539. i := index of pmsgIn^.V2.pszValueDN in valuesSeq
540. else
541. i := baseIndex
542. endif
543. j := 0
544. while (i < valuesSeq.length and j < 1000)
545. ls := LinkStamp(object, attr, valuesSeq[i])
546. if infoType = DS\_REPL\_INFO\_METADATA\_FOR\_ATTR\_VALUE then
547. pAttrValueMetaData :=
548. ADR(pmsgOut^.pAttrValueMetaData^.rgMetadata[j])
549. pAttrValueMetaData^.pszAttributeName := attr
550. pAttrValueMetaData^.pszObjectDN := object!distinguishedName
551. if (Syntax(attr) = Object(DN-Binary) or
552. Syntax(attr) = Object(DN-String)) then
553. pAttrValueMetaData^.cbData :=
554. length of data associated with valuesSeq[i]
555. pAttrValueMetaData^.pbData := data associated with
556. valuesSeq[i]
557. endif
558. pAttrValueMetaData^.ftimeCreated := ls.timeCreated
559. pAttrValueMetaData^.ftimeDeleted := ls.timeDeleted
560. pAttrValueMetaData^.dwVersion := ls.dwVersion
561. pAttrValueMetaData^.ftimeLastOriginatingChange :=
562. ls.timeChanged
563. pAttrValueMetaData^.uuidLastOriginatingDsaInvocationID :=
564. ls.uuidOriginating
565. pAttrValueMetaData^.usnOriginatingChange := ls.usnOriginating
566. pAttrValueMetaData^.usnLocalChange :=
567. implementation-specific value maintained for each link
568. attribute value
569. else
570. pAttrValueMetaData2 :=
571. ADR(pmsgOut^.pAttrValueMetaData2^.rgMetadata[j])
572. pAttrValueMetaData2^.pszAttributeName := attr
573. pAttrValueMetaData2^.pszObjectDN := object!distinguishedName
574. if (Syntax(attr) = Object(DN-Binary) or
575. Syntax(attr) = Object(DN-String)) then
576. pAttrValueMetaData2^.cbData :=
577. length of data associated with valuesSeq[i]
578. pAttrValueMetaData2^.pbData :=
579. data associated with valuesSeq[i]
580. endif
581. pAttrValueMetaData2^.ftimeCreated := ls.timeCreated
582. pAttrValueMetaData2^.ftimeDeleted := ls.timeDeleted
583. pAttrValueMetaData2^.dwVersion := ls.dwVersion
584. pAttrValueMetaData2^.ftimeLastOriginatingChange :=
585. ls.timeChanged
586. pAttrValueMetaData2^.uuidLastOriginatingDsaInvocationID :=
587. ls.uuidOriginating
588. pAttrValueMetaData2^.usnOriginatingChange :=
589. ls.usnOriginating
590. pAttrValueMetaData2^.usnLocalChange :=
591. implementation-specific value maintained for each
592. link attribute value
593. pAttrValueMetaData2^.pszLastOriginatingDsaDN :=
594. GetDNFromInvocationID(ls.uuidOriginating)
595. endif
596. i := i + 1
597. j := j + 1
598. endwhile
599. if infoType = DS\_REPL\_INFO\_METADATA\_FOR\_ATTR\_VALUE then
600. if i < valuesSeq.length - 1 then
601. /\* Since there are more entries to be returned, save the index
602. \* of the first value to be returned in the next call. \*/
603. pmsgOut^.pAttrValueMetaData^.dwEnumerationContext := i
604. else
605. /\* No more data is available. \*/
606. pmsgOut^.pAttrValueMetaData^.dwEnumerationContext :=
607. 0xffffffff
608. endif
609. pmsgOut^.pAttrValueMetaData^.cNumEntries = j
610. else
611. if i < valuesSeq.length - 1 then
612. /\* Since there are more entries to be returned, save the index
613. \* of the first value to be returned in the next call. \*/
614. pmsgOut^.pAttrValueMetaData2^.dwEnumerationContext := i
615. else
616. /\* No more data is available. \*/
617. pmsgOut^.pAttrValueMetaData2^.dwEnumerationContext :=
618. 0xffffffff
619. endif
620. pmsgOut^.pAttrValueMetaData2^.cNumEntries = j
621. endif
622. endif
623. endif
624. /\* DS\_REPL\_INFO\_KCC\_DSA\_CONNECT\_FAILURES \*/
625. if infoType = DS\_REPL\_INFO\_KCC\_DSA\_CONNECT\_FAILURES then
626. i := 0
627. foreach t in dc.kccFailedConnections
628. pConnectionFailure :=
629. ADR(pmsgOut^.pConnectionFailures^.rgDsaFailure[i])
630. pConnectionFailure^.pszDsaDN := t.DsaDN
631. pConnectionFailure^.uuidDsaObjGuid := t.UUIDDsa
632. pConnectionFailure^.fTimeFirstFailure := t.TimeFirstFailure
633. pConnectionFailure^.cNumFailures := t.FailureCount
634. pConnectionFailure^.dwLastResult := t.LastResult
635. i := i + 1
636. endfor
637. pmsgOut^.pConnectionFailures^.cNumEntries := i
638. endif
639. /\* DS\_REPL\_INFO\_KCC\_DSA\_LINK\_FAILURES \*/
640. if infoType = DS\_REPL\_INFO\_KCC\_DSA\_LINK\_FAILURES then
641. i := 0
642. foreach t in dc.kccFailedLinks
643. pConnectionLink := ADR(pmsgOut^.pLinkFailures^.rgDsaFailure[i])
644. pConnectionLink^.pszDsaDN := t.DsaDN
645. pConnectionLink^.uuidDsaObjGuid := t.UUIDDsa
646. pConnectionLink^.fTimeFirstFailure := t.TimeFirstFailure
647. pConnectionLink^.cNumFailures := t.FailureCount
648. pConnectionLink^.dwLastResult := t.LastResult
649. i := i + 1
650. endfor
651. pmsgOut^.pConnectionLinks^.cNumEntries := i
652. endif
653. /\* DS\_REPL\_INFO\_PENDING\_OPS \*/
654. if infoType = DS\_REPL\_INFO\_PENDING\_OPS then
655. i := 0
656. foreach t in dc.replicationQueue
657. pPendingOp := ADR(pmsgOut^.pPendingOps^.rgPendingOp[i])
658. pPendingOp^.fTimeEnqueued := t.TimeEnqueued
659. pPendingOp^.ulSerailNumber := t.SerialNumber
660. pPendingOp^.ulPriority := t.Priority
661. pPendingOp^.OpType := t.OperationType
662. pPendingOp^.ulOptions := t.Options
663. pPendingOp^.pszNamingContext := t.NamingContext
664. pPendingOp^.pszDsaDN := t.DsaDN
665. pPendingOp^.pszDsaAddress := t.DsaAddress
666. pPendingOp^.uuidNamingContextObjGuid := t.UUIDNC
667. pPendingOp^.uuidDsaObjGuid := t.UUIDDsa
668. i := i + 1
669. endfor
670. pmsgOut^.pPendingOps^.cNumPendingOps := i
671. pmsgOut^.pPendingOps^.fTimeCurrentOpStarted := time when current
672. operation was started
673. endif
674. /\* DS\_REPL\_INFO\_CLIENT\_CONTEXTS \*/
675. if infoType = DS\_REPL\_INFO\_CLIENT\_CONTEXTS then
676. i := 0
677. foreach t in dc.rpcClientContexts
678. pClientContext := ADR(pmsgOut^.pClientContexts^.rgContext[i])
679. pClientContext^.hCtx := t.BindingContext
680. pClientContext^.lReferenceCount := t.RefCount
681. pClientContext^.fIsBound := t.IsBound
682. pClientContext^.uuidClient := t.UUIDClient
683. pClientContext^.timeLastUsed := t.TimeLastUsed
684. pClientContext^.IPAddr := t.IPAddress
685. pClientContext^.pid := t.PID
686. i := i + 1
687. endfor
688. pmsgOut^.pClientContexts^.cNumContexts := i
689. endif
690. /\* DS\_REPL\_INFO\_SERVER\_OUTGOING\_CALLS \*/
691. if infoType = DS\_REPL\_INFO\_SERVER\_OUTGOING\_CALLS then
692. i := 0
693. foreach t in dc.rpcOutgoingContexts
694. pOutgoingContext =
695. ADR(pmsgOut^.pServerOutgoingCalls^.rgCall[i])
696. pOutgoingContext^.pszServerName := t.ServerName
697. pOutgoingContext^.fIsHandleBound := t.IsBound
698. pOutgoingContext^.fIsHandleFromCache := t.HandleFromCache
699. pOutgoingContext^.fIsHandleInCache := t.HandleInCache
700. pOutgoingContext^.dwThreadId := t.ThreadId
701. pOutgoingContext^.dwBindingTimeoutMins := t.BindingTimeOut
702. pOutgoingContext^.dstimeCreated := t.CreateTime
703. pOutgoingContext^.dwCallType := t.CallType
704. i := i + 1
705. endfor
706. pmsgOut^.pServerOutgoingCalls^.cNumCalls := i
707. endif
708. return 0

#### Examples of the IDL\_DRSGetReplInfo Method

##### Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_NEIGHBORS to find replication neighbors for a specified NC

In this example, the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) administrator wants to see which source [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) DC1 receives [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) from for the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) CONTOSO.COM. The domain administrator does so by issuing a request to DC1 with **pszObjectDN** set to the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the domain NC.

###### Initial State

Querying the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) CONTOSO.COM on DC1:

* ldap\_search\_s("DC=contoso,DC=com", baseObject, "(objectClass=domainDNS)", [objectClass, repsFrom])
* Getting 1 entry:
* Dn: DC=contoso,DC=com
  + 3> objectClass: top; domain; [**domainDNS**](#gt_646fb2d2-a783-4a48-832b-bd8491d54f1c);
  + 1> repsFrom: dwVersion: 2 v1.cb: 492 v1.cConsecutive Failures: 0 v1.timeLastSuccess: 12924402382 v1.timeLastAttempt: 12924402382 v1.ulResultLastAttempt: 0 v1.cbOtherDraOffset: 216v1.cbOtherDra: 276v1.ulReplicaFlags: 112 v1.rtSchedule: <skipped> v1.usnvec.usnHighObjUpdate: 19332 v1.usnvec.usnHighPropUpdate: 19332 v1.pszUuidDsaObj: 12626d52-1da7-4a40-a490-987c0880c3fe v1.pszUuidInvocId: 44a2959c-bb0d-4b2e-b106-fd8235288ee4 v1.pszUuidTransportObj: 00000000-0000-0000-0000-000000000000 v1.cbPASDataOffset: 0 v1~PasData: (none) v2~pdsa\_rpc\_inst v2.pszDSIServer 12626d52-1da7-4a40-a490-987c0880c3fe.\_msdcs.contoso.com v2.pszDSIAnnotation (null) v2.pszDSIInstance 12626d52-1da7-4a40-a490-987c0880c3fe.\_msdcs.contoso.com v2.pguidDSIInstance (null);

###### Client Request

The client invokes the [IDL\_DRSGetReplInfo (section 4.1.13)](#Section_dd29f9ceb30b411ebd54b77634eded47) method against DC1 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 2
* *pmsgIn* = DRS\_MSG\_GETREPLINFO\_REQ\_V2
  + InfoType = DS\_REPL\_INFO\_NEIGHBORS
  + pszObjectDN = "DC=contoso,DC=com"
  + uuidSourceDsaObjGuid = 00000000-0000-0000-0000-000000000000
  + ulFlags = 0x0
  + pszAttributeName = (null)
  + pszValueDN = (null)
  + dwEnumerationContext = 0

###### Server Response

A return code of 0 with the following values:

* *pdwOutVersion* = DS\_REPL\_INFO\_NEIGHBORS
* *pmsgOut* = DS\_REPL\_NEIGHBORSW
  + cNumNeighbors = 1
  + dwReserved = 0
  + rgNeighbor = DS\_REPL\_NEIGHBORW[]
    - rgNeighbor[0]
      * pszNamingContext = DC=contoso,dc=com
      * pszSourceDsaDN = CN=NTDS Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
      * pszSourceDsaAddress = 12626d52-1da7-4a40-a490-987c0880c3fe.\_msdcs.contoso.com
      * pszAsyncIntersiteTransportDN = (null)
      * dwReplicaFlags = 0x0
      * dwReserved = 0
      * uuidNamingContextObjGuid = 00000000-0000-0000-0000-000000000000
      * uuidSourceDsaObjGuid = 12626d52-1da7-4a40-a490-987c0880c3fe
      * uidSourceDsaInvocationID = 44a2959c-bb0d-4b2e-b106-fd8235288ee4
      * uuidAsyncIntersiteTransportObjGuid = 00000000-0000-0000-0000-000000000000
      * usnLastObjChangeSynced = 20002
      * usnAttributeFilter = 20002
      * ftimeLastSyncSuccess.dwLowDateTime = 0x4aaeb00
      * ftimeLastSyncSuccess.dwHighDateTime = 0x1cb2ad2
      * ftimeLastSyncAttempt.dwLowDateTime = 0x4aaeb00
      * ftimeLastSyncAttempt.dwHighDateTime = 0x1cb2ad2
      * dwLastSyncResult = 0
      * cNumConsecutiveSyncFailures = 0

###### Final State

The final state is the same as the initial state; there is no change.

##### Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_NEIGHBORS to find the naming contexts for which a DC receives updates from a replication neighbor

In this example, the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) administrator wants to see the [**NCs**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) for which DCA1 receives [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) from DC1. The domain administrator does so by issuing a request to DCA1 with **pszObjectDN** set to null and **uuidSourceDsaObjGuid** set to the [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of DC1.

###### Initial State

Querying the [**nTDSDSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8) for DC1 on DCA1:

* ldap\_search\_s("CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com", baseObject, "(objectClass=\*)", [objectClass, objectGUID])
* Getting 1 entry:
* >> Dn: CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + 3> objectClass: top; applicationSettings; nTDSDSA;
  + 1> objectGUID: 2e235fab-353c-46fc-8afd-437e9d0188b3;

Querying the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root object for config NC CN=Configuration,DC=contoso,DC=com on DCA1:

* ldap\_search\_s("CN=Configuration,DC=contoso,DC=com", baseObject, "(objectClass=\*)", [objectClass, repsFrom])
* Getting 1 entry:
* >> Dn: CN=Configuration,DC=contoso,DC=com
  + 2> objectClass: top; configuration;
  + 1> repsFrom: dwVersion: 2 v1.cb: 492 v1.cConsecutive Failures: 0 v1.timeLastSuccess: 12924750622 v1.timeLastAttempt: 12924750622 v1.ulResultLastAttempt: 0 v1.cbOtherDraOffset: 216v1.cbOtherDra: 276v1.ulReplicaFlags: 805306448 v1.rtSchedule: <skipped> v1.usnvec.usnHighObjUpdate: 24573 v1.usnvec.usnHighPropUpdate: 24573 v1.pszUuidDsaObj: 2e235fab-353c-46fc-8afd-437e9d0188b3 v1.pszUuidInvocId: 2e235fab-353c-46fc-8afd-437e9d0188b3 v1.pszUuidTransportObj: 00000000-0000-0000-0000-000000000000 v1.cbPASDataOffset: 0 v1~PasData: (none) v2~pdsa\_rpc\_inst v2.pszDSIServer 2e235fab-353c-46fc-8afd-437e9d0188b3.\_msdcs.contoso.com v2.pszDSIAnnotation (null) v2.pszDSIInstance 2e235fab-353c-46fc-8afd-437e9d0188b3.\_msdcs.contoso.com v2.pguidDSIInstance (null);

Querying the NC root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for [**schema NC**](#gt_754c2f7e-1fe4-4d1b-8976-6dad8d13e450) CN=Schema,CN=Configuration,DC=contoso,DC=com on DCA1:

* ldap\_search\_s("CN=Schema,CN=Configuration,DC=contoso,DC=com", baseObject, "(objectClass=\*)", [objectClass, repsFrom])
* Getting 1 entry:
* >> Dn: CN=Schema,CN=Configuration,DC=contoso,DC=com
  + 2> objectClass: top; dMD;
  + 1> repsFrom: dwVersion: 2 v1.cb: 492 v1.cConsecutive Failures: 0 v1.timeLastSuccess: 12924750622 v1.timeLastAttempt: 12924750622 v1.ulResultLastAttempt: 0 v1.cbOtherDraOffset: 216v1.cbOtherDra: 276v1.ulReplicaFlags: 2952790096 v1.rtSchedule: <skipped> v1.usnvec.usnHighObjUpdate: 24573 v1.usnvec.usnHighPropUpdate: 24573 v1.pszUuidDsaObj: 2e235fab-353c-46fc-8afd-437e9d0188b3 v1.pszUuidInvocId: 2e235fab-353c-46fc-8afd-437e9d0188b3 v1.pszUuidTransportObj: 00000000-0000-0000-0000-000000000000 v1.cbPASDataOffset: 0 v1~PasData: (none) v2~pdsa\_rpc\_inst v2.pszDSIServer 2e235fab-353c-46fc-8afd-437e9d0188b3.\_msdcs.contoso.com v2.pszDSIAnnotation (null) v2.pszDSIInstance 2e235fab-353c-46fc-8afd-437e9d0188b3.\_msdcs.contoso.com v2.pguidDSIInstance (null);

Querying the NC root object for NC DC=ForestDnsZones,DC=contoso,DC=com on DCA1:

* ldap\_search\_s("DC=ForestDnsZones,DC=contoso,DC=com ", baseObject, "(objectClass=\*)", [objectClass, repsFrom])
* Getting 1 entry:
* >> Dn: DC=ForestDnsZones,DC=contoso,DC=com
  + 3> objectClass: top; domain; domainDNS;
  + 1> repsFrom: dwVersion: 2 v1.cb: 492 v1.cConsecutive Failures: 0 v1.timeLastSuccess: 12924750622 v1.timeLastAttempt: 12924750622 v1.ulResultLastAttempt: 0 v1.cbOtherDraOffset: 216v1.cbOtherDra: 276v1.ulReplicaFlags: 805306448 v1.rtSchedule: <skipped> v1.usnvec.usnHighObjUpdate: 24573 v1.usnvec.usnHighPropUpdate: 24573 v1.pszUuidDsaObj: 2e235fab-353c-46fc-8afd-437e9d0188b3 v1.pszUuidInvocId: 2e235fab-353c-46fc-8afd-437e9d0188b3 v1.pszUuidTransportObj: 00000000-0000-0000-0000-000000000000 v1.cbPASDataOffset: 0 v1~PasData: (none) v2~pdsa\_rpc\_inst v2.pszDSIServer 2e235fab-353c-46fc-8afd-437e9d0188b3.\_msdcs.contoso.com v2.pszDSIAnnotation (null) v2.pszDSIInstance 2e235fab-353c-46fc-8afd-437e9d0188b3.\_msdcs.contoso.com v2.pguidDSIInstance (null);

###### Client Request

The client invokes the [IDL\_DRSGetReplInfo (section 4.1.13)](#Section_dd29f9ceb30b411ebd54b77634eded47) method against DCA1 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 2
* *pmsgIn* = DRS\_MSG\_GETREPLINFO\_REQ\_V2
  + InfoType = DS\_REPL\_INFO\_NEIGHBORS
  + pszObjectDN = (null)
  + uuidSourceDsaObjGuid = 2e235fab-353c-46fc-8afd-437e9d0188b3
  + ulFlags = 0x0
  + pszAttributeName = (null)
  + pszValueDN = (null)
  + dwEnumerationContext = 0

###### Server Response

A return code of 0 with the following values:

* *pdwOutVersion* = DS\_REPL\_INFO\_NEIGHBORS
* *pmsgOut* = DS\_REPL\_NEIGHBORSW
  + cNumNeighbors = 3
  + dwReserved = 0
  + rgNeighbor = DS\_REPL\_NEIGHBORW[]
  + rgNeighbor[0]
    - pszNamingContext = CN=Configuration,DC=contoso,DC=com
    - pszSourceDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
    - pszSourceDsaAddress = 2e235fab-353c-46fc-8afd-437e9d0188b3.\_msdcs.contoso.com
    - pszAsyncIntersiteTransportDN = (null)
    - dwReplicaFlags = 0x0
    - dwReserved = 0
    - uuidNamingContextObjGuid = 64f4ed75-28b1-42f3-b7c9-6ac234db9a9e
    - uuidSourceDsaObjGuid = 2e235fab-353c-46fc-8afd-437e9d0188b3
    - uidSourceDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
    - uuidAsyncIntersiteTransportObjGuid = 00000000-0000-0000-0000-000000000000
    - usnLastObjChangeSynced = 24523
    - usnAttributeFilter = 24523
    - ftimeLastSyncSuccess.dwLowDateTime = 0xf7e80900
    - ftimeLastSyncSuccess.dwHighDateTime = 0x1cb2de9
    - ftimeLastSyncAttempt.dwLowDateTime = 0xf7e80900
    - ftimeLastSyncAttempt.dwHighDateTime = 0x1cb2de9
    - dwLastSyncResult = 0
    - cNumConsecutiveSyncFailures = 0
  + rgNeighbor[1]
    - pszNamingContext = CN=Schema,CN=Configuration,DC=contoso,DC=com
    - pszSourceDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
    - pszSourceDsaAddress = 2e235fab-353c-46fc-8afd-437e9d0188b3.\_msdcs.contoso.com
    - pszAsyncIntersiteTransportDN = (null)
    - dwReplicaFlags = 0x0
    - dwReserved = 0
    - uuidNamingContextObjGuid = f3ba2060-2d67-43e6-a334-54a8f1ecc78a
    - uuidSourceDsaObjGuid = 2e235fab-353c-46fc-8afd-437e9d0188b3
    - uidSourceDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
    - uuidAsyncIntersiteTransportObjGuid = 00000000-0000-0000-0000-000000000000
    - usnLastObjChangeSynced = 24523
    - usnAttributeFilter = 24523
    - ftimeLastSyncSuccess.dwLowDateTime = 0xf7e80900
    - ftimeLastSyncSuccess.dwHighDateTime = 0x1cb2de9
    - ftimeLastSyncAttempt.dwLowDateTime = 0xf7e80900
    - ftimeLastSyncAttempt.dwHighDateTime = 0x1cb2de9
    - dwLastSyncResult = 0
    - cNumConsecutiveSyncFailures = 0
  + rgNeighbor[2]
    - pszNamingContext = DC=ForestDnsZones,DC=contoso,DC=com
    - pszSourceDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
    - pszSourceDsaAddress = 2e235fab-353c-46fc-8afd-437e9d0188b3.\_msdcs.contoso.com
    - pszAsyncIntersiteTransportDN = (null)
    - dwReplicaFlags = 0x0
    - dwReserved = 0
    - uuidNamingContextObjGuid = 7fafd728-d866-4cf3-915f-78ff680603d4
    - uuidSourceDsaObjGuid = 2e235fab-353c-46fc-8afd-437e9d0188b3
    - uidSourceDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
    - uuidAsyncIntersiteTransportObjGuid = 00000000-0000-0000-0000-000000000000
    - usnLastObjChangeSynced = 24523
    - usnAttributeFilter = 24523
    - ftimeLastSyncSuccess.dwLowDateTime = 0xf7e80900
    - ftimeLastSyncSuccess.dwHighDateTime = 0x1cb2de9
    - ftimeLastSyncAttempt.dwLowDateTime = 0xf7e80900
    - ftimeLastSyncAttempt.dwHighDateTime = 0x1cb2de9
    - dwLastSyncResult = 0
    - cNumConsecutiveSyncFailures = 0

###### Final State

The final state is the same as the initial state; there is no change.

##### Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_REPSTO to find replication neighbors for a specified NC

In this example, the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) administrator verifies which [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) DC1 sends [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) to for the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) CONTOSO.COM. The domain administrator does this by issuing a request to DC1 with **pszObjectDN** set to the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the domain NC.

###### Initial State

Querying the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) CONTOSO.COM on DC1:

* ldap\_search\_s("DC=contoso,DC=com", baseObject, "(objectClass=domainDNS)", [objectClass, repsTo])
* Getting 1 entry:
* >> Dn: DC=contoso,DC=com
  + 3> objectClass: top; domain; domainDNS;
  + 1> repsTo: dwVersion: 2 v1.cb: 492 v1.cConsecutive Failures: 0 v1.timeLastSuccess: 12924828300 v1.timeLastAttempt: 12924828300 v1.ulResultLastAttempt: 0 v1.cbOtherDraOffset: 216v1.cbOtherDra: 276v1.ulReplicaFlags: 16 v1.rtSchedule: <skipped> v1.usnvec.usnHighObjUpdate: 0 v1.usnvec.usnHighPropUpdate: 0 v1.pszUuidDsaObj: 12626d52-1da7-4a40-a490-987c0880c3fe v1.pszUuidInvocId: 00000000-0000-0000-0000-000000000000 v1.pszUuidTransportObj: 00000000-0000-0000-0000-000000000000 v1.cbPASDataOffset: 0 v1~PasData: (none) v2~pdsa\_rpc\_inst v2.pszDSIServer 12626d52-1da7-4a40-a490-987c0880c3fe.\_msdcs.contoso.com v2.pszDSIAnnotation (null) v2.pszDSIInstance 12626d52-1da7-4a40-a490-987c0880c3fe.\_msdcs.contoso.com v2.pguidDSIInstance (null);

###### Client Request

The client invokes the [IDL\_DRSGetReplInfo (section 4.1.13)](#Section_dd29f9ceb30b411ebd54b77634eded47) method against DC1 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 2
* *pmsgIn* = DRS\_MSG\_GETREPLINFO\_REQ\_V2
  + InfoType = DS\_REPL\_INFO\_REPSTO
  + pszObjectDN = DC=contoso,DC=com
  + uuidSourceDsaObjGuid = 00000000-0000-0000-0000-000000000000
  + ulFlags = 0x0
  + pszAttributeName = (null)
  + pszValueDN = (null)
  + dwEnumerationContext = 0

###### Server Response

A return code of 0 with the following values:

* *pdwOutVersion* = DS\_REPL\_INFO\_REPSTO
* *pmsgOut* = DS\_REPL\_NEIGHBORSW
  + cNumNeighbors = 1
  + dwReserved = 0
  + rgNeighbor = DS\_REPL\_NEIGHBORW[]
    - rgNeighbor[0]
      * pszNamingContext = DC=contoso,DC=com
      * pszSourceDsaDN = CN=NTDS Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
      * pszSourceDsaAddress = 12626d52-1da7-4a40-a490-987c0880c3fe.\_msdcs.contoso.com
      * pszAsyncIntersiteTransportDN = (null)
      * dwReplicaFlags = 0x0
      * dwReserved = 0
      * uuidNamingContextObjGuid = 00000000-0000-0000-0000-000000000000
      * uuidSourceDsaObjGuid = 12626d52-1da7-4a40-a490-987c0880c3fe
      * uidSourceDsaInvocationID = 00000000-0000-0000-0000-000000000000
      * uuidAsyncIntersiteTransportObjGuid = 00000000-0000-0000-0000-000000000000
      * usnLastObjChangeSynced = 0
      * usnAttributeFilter = 0
      * ftimeLastSyncSuccess.dwLowDateTime = 0x6a6bee00
      * ftimeLastSyncSuccess.dwHighDateTime = 0x1cb2ea9
      * ftimeLastSyncAttempt.dwLowDateTime = 0x6a6bee00
      * ftimeLastSyncAttempt.dwHighDateTime = 0x1cb2ea9
      * dwLastSyncResult = 0
      * cNumConsecutiveSyncFailures = 0

###### Final State

The final state is the same as the initial state; there is no change.

##### Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_CURSORS\_3\_FOR\_NC

In this example, the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) administrator wants to view the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state on DC2 relative to the [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625) CN=Configuration,DC=contoso,DC=com. The domain administrator does this by issuing a request to DC1 with **pszObjectDN** set to the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the config NC.

###### Initial State

Querying the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for the [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625) CN=Configuration,DC=contoso,DC=com on DC2:

* ldap\_search\_s("CN=Configuration,DC=contoso,DC=com", baseObject, "(objectClass=configuration)", [objectClass, replUpToDateVector])
* Getting 1 entry:
* >> Dn: CN=Configuration,DC=contoso,DC=com
  + 2> objectClass: top; configuration;
  + 1> replUpToDateVector: dwVersion: 2, dwReserved1: 0, V2.cNumCursors: 2, V2.dwReserved2: 0, rgCursors: {uuidDsa: 2e235fab-353c-46fc-8afd-437e9d0188b3, usnHighPropUpdate: 22378, timeLastSyncSuccess: 07/26/2010 16:00:19}, {uuidDsa: e4dfc4c0-381c-48c9-a563-cb27db448753, usnHighPropUpdate: 18177, timeLastSyncSuccess: 07/26/2010 16:02:32};

###### Client Request

The client invokes the [IDL\_DRSGetReplInfo (section 4.1.13)](#Section_dd29f9ceb30b411ebd54b77634eded47) method against DC2 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC2 omitted):

* *dwInVersion* = 2
* *pmsgIn* = DRS\_MSG\_GETREPLINFO\_REQ\_V2
  + InfoType = DS\_REPL\_INFO\_CURSORS\_3\_FOR\_NC
  + pszObjectDN = CN=Configuration,DC=contoso,dc=com
  + uuidSourceDsaObjGuid = 00000000-0000-0000-0000-000000000000
  + ulFlags = 0x0
  + pszAttributeName = (null)
  + pszValueDN = (null)
  + dwEnumerationContext = 0

###### Server Response

A return code of 0 with the following values:

* *pdwOutVersion* = DS\_REPL\_INFO\_CURSORS\_3\_FOR\_NC
* *pmsgOut* = DS\_REPL\_CURSORS\_3W
  + cNumCursors = 3
  + dwEnumerationContext = 0xffffffff
  + rgCursor = DS\_REPL\_CURSOR\_3W[]
    - rgCursor[0]
      * usnAttributeFilter = 22378
      * uuidSourceDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
      * ftimeLastSyncSuccess.dwLowDateTime = 0x517f0380
      * ftimeLastSyncSuccess.dwHighDateTime = 0x1cb2d16
      * pszSourceDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
    - rgCursor[1]
      * usnAttributeFilter = 43601
      * uuidSourceDsaInvocationID = 44a2959c-bb0d-4b2e-b106-fd8235288ee4
      * ftimeLastSyncSuccess.dwLowDateTime = 0xaf135000
      * ftimeLastSyncSuccess.dwHighDateTime = 0x1cb2d16
      * pszSourceDsaDN = CN=NTDS Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
    - rgCursor[2]
      * usnAttributeFilter = 18177
      * uuidSourceDsaInvocationID = e4dfc4c0-381c-48c9-a563-cb27db448753
      * ftimeLastSyncSuccess.dwLowDateTime = 0xa0c53400
      * ftimeLastSyncSuccess.dwHighDateTime = 0x1cb2d16
      * pszSourceDsaDN = CN=NTDS Settings,CN=DCA1,CN=Servers,CN=Default-Second-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com

###### Final State

The final state is the same as the initial state; there is no change.

##### Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_METADATA\_2\_FOR\_OBJ

In this example, the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) administrator wants to view the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state of the organizational unit OU1 on DC1.

###### Initial State

Querying OU1 on DC1:

* ldap\_search\_s("OU=OU1,DC=contoso,DC=com", baseObject, "(objectClass=\*)", [\*])
* Getting 1 entry:
* >> Dn: OU=OU1,DC=contoso,DC=com
  + 1> distinguishedName: OU=OU1,DC=contoso,DC=com;
  + 3> dSCorePropagationData: 7/27/2010 10:20:24 PM Pacific Daylight Time; 7/27/2010 10:20:23 PM Pacific Daylight Time; 0x0 = ( ), 0x0 = ( );
  + 1> instanceType: 0x4 = ( WRITE );
  + 1> name:OU1;
  + 1> objectCategory: CN=Organizational-Unit,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; organizationalUnit;
  + 1> objectGUID: 1a0c2e8f-2747-4e38-80fb-074e2dd3df8c;
  + 1> ou: OU1;
  + 1> uSNChanged: 25426;
  + 1> uSNCreated: 25424;
  + 1> whenChanged: 7/27/2010 10:20:23 PM Pacific Daylight Time;
  + 1> whenCreated: 7/27/2010 10:20:23 PM Pacific Daylight Time;

###### Client Request

The client invokes the [IDL\_DRSGetReplInfo (section 4.1.13)](#Section_dd29f9ceb30b411ebd54b77634eded47) method against DC2 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 2
* *pmsgIn* = DRS\_MSG\_GETREPLINFO\_REQ\_V2
  + InfoType = DS\_REPL\_INFO\_METADATA\_2\_FOR\_OBJ
  + pszObjectDN = OU=OU1,DC=contoso,dc=com
  + uuidSourceDsaObjGuid = 00000000-0000-0000-0000-000000000000
  + ulFlags = 0x0
  + pszAttributeName = (null)
  + pszValueDN = (null)
  + dwEnumerationContext = 0

###### Server Response

A return code of 0 with the following values:

* *pdwOutVersion* = DS\_REPL\_INFO\_METADATA\_2\_FOR\_OBJ
* *pmsgOut* = DS\_REPL\_OBJ\_META\_DATA\_2
  + cNumEntries = 7
  + dwReserved = 0
  + rgMetaData = DS\_REPL\_ATTR\_META\_DATA\_2[]
    - rgMetaData[0]
      * pszAttributeName = objectClass
      * dwVersion = 1
      * ftimeLastOriginatingChange.dwLowDateTime = 0x94270580
      * ftimeLastOriginatingChange.dwHighDateTime = 0x1cb2e14
      * uuidLastOriginatingDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
      * usnOriginatingChange = 25424
      * usnLocalChange = 25424
      * pszLastOriginatingDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
    - rgMetaData[1]
      * pszAttributeName = ou
      * dwVersion = 1
      * ftimeLastOriginatingChange.dwLowDateTime = 0x94270580
      * ftimeLastOriginatingChange.dwHighDateTime = 0x1cb2e14
      * uuidLastOriginatingDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
      * usnOriginatingChange = 25424
      * usnLocalChange = 25424
      * pszLastOriginatingDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
    - rgMetaData[2]
      * pszAttributeName = instanceType
      * dwVersion = 1
      * ftimeLastOriginatingChange.dwLowDateTime = 0x94270580
      * ftimeLastOriginatingChange.dwHighDateTime = 0x1cb2e14
      * uuidLastOriginatingDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
      * usnOriginatingChange = 25424
      * usnLocalChange = 25424
      * pszLastOriginatingDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
    - rgMetaData[3]
      * pszAttributeName = whenCreated
      * dwVersion = 1
      * ftimeLastOriginatingChange.dwLowDateTime = 0x94270580
      * ftimeLastOriginatingChange.dwHighDateTime = 0x1cb2e14
      * uuidLastOriginatingDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
      * usnOriginatingChange = 25424
      * usnLocalChange = 25424
      * pszLastOriginatingDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
    - rgMetaData[4]
      * pszAttributeName = nTSecurityDescriptor
      * dwVersion = 2
      * ftimeLastOriginatingChange.dwLowDateTime = 0x94270580
      * ftimeLastOriginatingChange.dwHighDateTime = 0x1cb2e14
      * uuidLastOriginatingDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
      * usnOriginatingChange = 25426
      * usnLocalChange = 25426
      * pszLastOriginatingDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
    - rgMetaData[5]
      * pszAttributeName = name
      * dwVersion = 1
      * ftimeLastOriginatingChange.dwLowDateTime = 0x94270580
      * ftimeLastOriginatingChange.dwHighDateTime = 0x1cb2e14
      * uuidLastOriginatingDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
      * usnOriginatingChange = 25424
      * usnLocalChange = 25424
      * pszLastOriginatingDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
    - rgMetaData[6]
      * pszAttributeName = objectCategory
      * dwVersion = 1
      * ftimeLastOriginatingChange.dwLowDateTime = 0x94270580
      * ftimeLastOriginatingChange.dwHighDateTime = 0x1cb2e14
      * uuidLastOriginatingDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
      * usnOriginatingChange = 25424
      * usnLocalChange = 25424
      * pszLastOriginatingDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com

###### Final State

The final state is the same as the initial state; there is no change.

##### Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_METADATA\_2\_FOR\_ATTR\_VALUE to view the replication metadata for all values of a link value attribute

In this example, the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) administrator requires viewing the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state of the [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) member of the [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) GroupB on DC2.

###### Initial State

Querying the GroupB on DC2:

* ldap\_search\_s("CN=GroupB,CN=Users,DC=contoso,DC=com", baseObject, "(objectClass=group)", [objectClass, member])
* Getting 1 entry:
* >> Dn: CN=GroupB,CN=Users,DC=contoso,DC=com
  + 3> member: CN=GroupC,CN=Users,DC=contoso,DC=com; CN=GroupA,CN=Users,DC=contoso,DC=com; CN=Kim Akers,CN=Users,DC=contoso,DC=com;
  + 2> objectClass: top; group;

###### Client Request

The client invokes the [IDL\_DRSGetReplInfo (section 4.1.13)](#Section_dd29f9ceb30b411ebd54b77634eded47) method against DC2 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC2 omitted):

* *dwInVersion* = 2
* *pmsgIn* = DRS\_MSG\_GETREPLINFO\_REQ\_V2
  + InfoType = DS\_REPL\_INFO\_METADATA\_2\_FOR\_ATTR\_VALUE
  + pszObjectDN = CN=GroupB,CN=Users,DC=contoso,dc=com
  + uuidSourceDsaObjGuid = 00000000-0000-0000-0000-000000000000
  + ulFlags = 0x0
  + pszAttributeName = member
  + pszValueDN = (null)
  + dwEnumerationContext = 0

###### Server Response

A return code of 0 with the following values:

* *pdwOutVersion* = DS\_REPL\_INFO\_METADATA\_2\_FOR\_ATTR\_VALUE
* *pmsgOut* = DS\_REPL\_ATTR\_VALUE\_META\_DATA\_2
  + cNumEntries = 3
  + dwEnumerationContext = 0xffffffff
  + rgMetaData = DS\_REPL\_VALUE\_META\_DATA\_2[]
    - rgMetaData[0]
      * pszAttributeName = member
      * pszObjectDn = CN=Kim Akers,CN=Users,DC=contoso,DC=com
      * cbData = 0
      * pbData = null
      * ftimeDeleted.dwLowDateTime = 0x0
      * ftimeDeleted.dwHighDateTime = 0x0
      * ftimeCreated.dwLowDateTime = 0xc3a4dd80
      * ftimeCreated.dwHighDateTime = 0x1cb2ab5
      * dwVersion = 1
      * ftimeLastOriginatingChange.dwLowDateTime = 0xc3a4dd80
      * ftimeLastOriginatingChange.dwHighDateTime = 0x1cb2ab5
      * uuidLastOriginatingDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
      * usnOriginatingChange = 15399
      * usnLocalChange = 19212
      * pszLastOriginatingDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
    - rgMetaData[1]
      * pszAttributeName = member
      * pszObjectDn = CN=GroupA,CN=Users,DC=contoso,DC=com
      * cbData = 0
      * pbData = null
      * ftimeDeleted.dwLowDateTime = 0x0
      * ftimeDeleted.dwHighDateTime = 0x0
      * ftimeCreated.dwLowDateTime = 0x2fb77680
      * ftimeCreated.dwHighDateTime = 0x1cb2e13
      * dwVersion = 1
      * ftimeLastOriginatingChange.dwLowDateTime = 0x2fb77680
      * ftimeLastOriginatingChange.dwHighDateTime = 0x1cb2e13
      * uuidLastOriginatingDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
      * usnOriginatingChange = 25384
      * usnLocalChange = 46509
      * pszLastOriginatingDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
    - rgMetaData[2]
      * pszAttributeName = member
      * pszObjectDn = CN=GroupC,CN=Users,DC=contoso,DC=com
      * cbData = 0
      * pbData = null
      * ftimeDeleted.dwLowDateTime = 0x0
      * ftimeDeleted.dwHighDateTime = 0x0
      * ftimeCreated.dwLowDateTime = 0x2fb77680
      * ftimeCreated.dwHighDateTime = 0x1cb2e13
      * dwVersion = 1
      * ftimeLastOriginatingChange.dwLowDateTime = 0x2fb77680
      * ftimeLastOriginatingChange.dwHighDateTime = 0x1cb2e13
      * uuidLastOriginatingDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
      * usnOriginatingChange = 25385
      * usnLocalChange = 46508
      * pszLastOriginatingDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com

###### Final State

The final state is the same as the initial state; there is no change.

##### Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_METADATA\_2\_FOR\_ATTR\_VALUE to view the replication metadata for a specific value of a link value attribute

In this example, the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) administrator wants to view the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state of [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) member of the [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) GroupB on DC2. Specifically, the domain administrator is interested in the member value corresponding to Kim Akers' account. The domain administrator does so by issuing a request to DC2 with **pszObjectDN** set to the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of GroupB, **pszAttributeName** set to member, and **pszValueDN** set to the DN of Kim Akers' user account.

###### Initial State

Querying the GroupB on DC2:

* ldap\_search\_s("CN=GroupB,CN=Users,DC=contoso,DC=com", baseObject, "(objectClass=group)", [objectClass, member])
* Getting 1 entry:
* >> Dn: CN=GroupB,CN=Users,DC=contoso,DC=com
  + 3> member: CN=GroupC,CN=Users,DC=contoso,DC=com; CN=GroupA,CN=Users,DC=contoso,DC=com; CN=Kim Akers,CN=Users,DC=contoso,DC=com;
  + 2> objectClass: top; group;

###### Client Request

The client invokes the [IDL\_DRSGetReplInfo (section 4.1.13)](#Section_dd29f9ceb30b411ebd54b77634eded47) method against DC2 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC2 omitted):

* *dwInVersion* = 2
* *pmsgIn* = DRS\_MSG\_GETREPLINFO\_REQ\_V2
  + InfoType = DS\_REPL\_INFO\_METADATA\_2\_FOR\_ATTR\_VALUE
  + pszObjectDN = CN=GroupB,CN=Users,DC=contoso,dc=com
  + uuidSourceDsaObjGuid = 00000000-0000-0000-0000-000000000000
  + ulFlags = 0x0
  + pszAttributeName = member
  + pszValueDN = CN=Kim Akers,CN=Users,DC=contoso,DC=com
  + dwEnumerationContext = 0

###### Server Response

A return code of 0 with the following values:

* *pdwOutVersion* = DS\_REPL\_INFO\_METADATA\_2\_FOR\_ATTR\_VALUE
* *pmsgOut* = DS\_REPL\_ATTR\_VALUE\_META\_DATA\_2
  + cNumEntries = 1
  + dwEnumerationContext = 0x1
  + rgMetaData = DS\_REPL\_VALUE\_META\_DATA\_2[]
    - rgMetaData[0]
      * pszAttributeName = member
      * pszObjectDn = CN=Kim Akers,CN=Users,DC=contoso,DC=com
      * cbData = 0
      * pbData = null
      * ftimeDeleted.dwLowDateTime = 0x0
      * ftimeDeleted.dwHighDateTime = 0x0
      * ftimeCreated.dwLowDateTime = 0xc3a4dd80
      * ftimeCreated.dwHighDateTime = 0x1cb2ab5
      * dwVersion = 1
      * ftimeLastOriginatingChange.dwLowDateTime = 0xc3a4dd80
      * ftimeLastOriginatingChange.dwHighDateTime = 0x1cb2ab5
      * uuidLastOriginatingDsaInvocationID = 2e235fab-353c-46fc-8afd-437e9d0188b3
      * usnOriginatingChange = 15399
      * usnLocalChange = 19212
      * pszLastOriginatingDsaDN = CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com

###### Final State

The final state is the same as the initial state; there is no change.

##### Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_KCC\_DSA\_CONNECT\_FAILURES

In this example, the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) administrator verifies whether DC1 has any [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) failures.

###### Initial State

DC2 is a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) neighbor of DC1. DC2 is offline and DC1 is unable to contact DC2 to query its replication state.

###### Client Request

The client invokes the [IDL\_DRSGetReplInfo (section 4.1.13)](#Section_dd29f9ceb30b411ebd54b77634eded47) method against DC1 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 2
* *pmsgIn* = DRS\_MSG\_GETREPLINFO\_REQ\_V2
  + InfoType = DS\_REPL\_INFO\_KCC\_DSA\_CONNECT\_FAILURES
  + pszObjectDN = (null)
  + uuidSourceDsaObjGuid = 00000000-0000-0000-0000-000000000000
  + ulFlags = 0x0
  + pszAttributeName = (null)
  + pszValueDN = (null)
  + dwEnumerationContext = 0

###### Server Response

A return code of 0 with the following values:

* *pdwOutVersion* = DS\_REPL\_INFO\_KCC\_DSA\_CONNECT\_FAILURES
* *pmsgOut* = DS\_REPL\_KCC\_DSA\_FAILURESW
  + cNumEntries = 1
  + dwReserved = 0
  + rgDsaFailure = DS\_REPL\_KCC\_DSA\_FAILUREW[]
    - rgDsaFailure[0]
      * pszDsaDN = CN=NTDS Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
      * uuidDsaObjGuid = 12626d52-1da7-4a40-a490-987c0880c3fe
      * ftimeFirstFailure.dwLowDateTime = 0xcefc4100
      * ftimeFirstFailure.dwHighDateTime = 0x1cb2d21
      * cNumFailures = 2
      * dwLastResult = 1722

###### Final State

The final state is the same as the [initial state (section 4.1.13.4.8.1)](#Section_aeaaad17338c4569b9044aa5aef3b01e); there is no change.

##### Calling IDL\_DRSGetReplInfo with infoType DS\_REPL\_INFO\_PENDING\_OPS

In this example, the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) administrator verifies whether DC1 has any pending operations in its [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) queue.

###### Initial State

DC2 is a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) neighbor of DC1. DC1 is syncing [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) from DC2.

###### Client Request

The client invokes the [IDL\_DRSGetReplInfo (section 4.1.13)](#Section_dd29f9ceb30b411ebd54b77634eded47) method against DC1 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 2
  + *pmsgIn* = DRS\_MSG\_GETREPLINFO\_REQ\_V2
    - InfoType = DS\_REPL\_INFO\_PENDING\_OPS
    - pszObjectDN = (null)
    - uuidSourceDsaObjGuid = 00000000-0000-0000-0000-000000000000
    - ulFlags = 0x0
    - pszAttributeName = (null)
    - pszValueDN = (null)
    - dwEnumerationContext = 0

###### Server Response

A return code of 0 with the following values:

* *pdwOutVersion* = DS\_REPL\_INFO\_PENDING\_OPS
* *pmsgOut* = DS\_REPL\_PENDING\_OPSW
  + ftimeCurrentOpStarted.dwLowDateTime = 0x2546bc80
  + ftimeCurrentOpStarted.dwHighDateTime = 0x1cb2ea7
  + cNumPendingOps = 1
  + rgPendingOp = DS\_REPL\_OPW[]
    - rgPendingOp[0]
      * ftimeEnqueued.dwLowDateTime = 0x2546bc80
      * ftimeEnqueued.dwHighDateTime = 0x2546bc80
      * ulSerialNumber = 2343
      * ulPriority = 250
      * OpType = DS\_REPL\_OP\_TYPE\_SYNC
      * ulOptions = 524291
      * pszNamingContext = DC=contoso,DC=com
      * pszDsaDN = CN=NTDS Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
      * pszDsaAddress = 12626d52-1da7-4a40-a490-987c0880c3fe.\_msdcs.contoso.com
      * uuidNamingContextObjGuid = 8f3cea57-61ff-46cb-aa17-6c1683c33020
      * uuidDsaObjGuid = 12626d52-1da7-4a40-a490-987c0880c3fe

###### Final State

The final state is the same as the [initial state (section 4.1.13.4.9.1)](#Section_4671f3017be54801b1e201c174f34428); there is no change.

### IDL\_DRSInitDemotion (Opnum 25)

The IDL\_DRSInitDemotion method performs the first phase of the removal of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) from an [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). This method is supported only by AD LDS.

1. ULONG IDL\_DRSInitDemotion(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_INIT\_DEMOTIONREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_INIT\_DEMOTIONREPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_INIT\_DEMOTIONREQ

The DRS\_MSG\_INIT\_DEMOTIONREQ union defines request messages sent to the [IDL\_DRSInitDemotion](#Section_faca17da3f7f409298dbfd2ce7d98b8c) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_INIT\_DEMOTIONREQ\_V1 V1;
6. } DRS\_MSG\_INIT\_DEMOTIONREQ;

**V1:**  Version 1 request. Currently, only one version is defined.

##### DRS\_MSG\_INIT\_DEMOTIONREQ\_V1

The DRS\_MSG\_INIT\_DEMOTIONREQ\_V1 structure defines a request message sent to the [IDL\_DRSInitDemotion](#Section_faca17da3f7f409298dbfd2ce7d98b8c) method.

1. typedef struct {
2. DWORD dwReserved;
3. } DRS\_MSG\_INIT\_DEMOTIONREQ\_V1;

**dwReserved:**  Unused. MUST be 0.

##### DRS\_MSG\_INIT\_DEMOTIONREPLY

The DRS\_MSG\_INIT\_DEMOTIONREPLY union defines the response messages received from the [IDL\_DRSInitDemotion](#Section_faca17da3f7f409298dbfd2ce7d98b8c) method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_INIT\_DEMOTIONREPLY\_V1 V1;
6. } DRS\_MSG\_INIT\_DEMOTIONREPLY;

**V1:**  Version 1 reply.

##### DRS\_MSG\_INIT\_DEMOTIONREPLY\_V1

The DRS\_MSG\_INIT\_DEMOTIONREPLY\_V1 structure defines a response message received from the [IDL\_DRSInitDemotion](#Section_faca17da3f7f409298dbfd2ce7d98b8c) method.

1. typedef struct {
2. DWORD dwOpError;
3. } DRS\_MSG\_INIT\_DEMOTIONREPLY\_V1;

**dwOpError:**  A Win32 error code, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.2.

#### Server Behavior of the IDL\_DRSInitDemotion Method

*Informative summary of behavior*: Performs the first phase of the removal of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) from an [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). This phase consists of disabling both originating and [**replicated updates**](#gt_2a923099-db0a-4932-af28-4354601e85c4) to the AD LDS DC.[<33>](#Appendix_A_33" \o "Product behavior note 33)

1. ULONG
2. IDL\_DRSInitDemotion(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_INIT\_DEMOTIONREQ\* pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DRS\_MSG\_INIT\_DEMOTIONREPLY\* pmsgOut
10. )
11. msgIn: DRS\_MSG\_INIT\_DEMOTIONREQ\_V1
12. ret: DWORD
13. ValidateDRSInput(hDrs, 25)
14. pmsgOut^.V1.dwOpError := 0
15. if dwInVersion ≠ 1 then
16. return ERROR\_INVALID\_PARAMETER
17. endif
18. if pmsgIn = null then
19. return ERROR\_INVALID\_PARAMETER
20. endif
21. if pmsgIn^.V1.dwReserved ≠ 0 then
22. return ERROR\_INVALID\_PARAMETER
23. endif
24. msgIn := pmsgIn^.V1
25. if not IsMemberOfBuiltinAdminGroup() then
26. /\* only BA is allowed to demote an AD LDS service \*/
27. return ERROR\_DS\_DRA\_ACCESS\_DENIED
28. endif
29. dc.fEnableUpdates := FALSE
30. pmsgOut^.V1.dwOpError := ERROR\_SUCCESS
31. pdwOutVersion^ := 1
32. return ERROR\_SUCCESS

### IDL\_DRSInterDomainMove (Opnum 10)

The IDL\_DRSInterDomainMove method is a helper method used in a cross-[**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) move [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) operation.

1. ULONG IDL\_DRSInterDomainMove(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_MOVEREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_MOVEREPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_MOVEREQ

The DRS\_MSG\_MOVEREQ union defines the request messages sent to the [IDL\_DRSInterDomainMove](#Section_595b2fef493b4b1db60de7a1a3345b0e) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_MOVEREQ\_V1 V1;
6. [case(2)]
7. DRS\_MSG\_MOVEREQ\_V2 V2;
8. } DRS\_MSG\_MOVEREQ;

**V1:**  The version 1 request (obsolete).

**V2:**  The version 2 request.

##### DRS\_MSG\_MOVEREQ\_V1

The DRS\_MSG\_MOVEREQ\_V1 structure defines a request message sent to the [IDL\_DRSInterDomainMove](#Section_595b2fef493b4b1db60de7a1a3345b0e) method. This request version is obsolete.[<34>](#Appendix_A_34" \o "Product behavior note 34)

1. typedef struct {
2. char\* pSourceDSA;
3. ENTINF\* pObject;
4. UUID\* pParentUUID;
5. SCHEMA\_PREFIX\_TABLE PrefixTable;
6. ULONG ulFlags;
7. } DRS\_MSG\_MOVEREQ\_V1;

**pSourceDSA:**  The [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) of the client [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**pObject:**  The [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to be moved.

**pParentUUID:**  The objectGUID of the new [**parent object**](#gt_0d41951a-62f0-4fbd-bb23-22f645ae3bf5).

**PrefixTable:**  The [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) with which to translate the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values in **pObject** to [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

**ulFlags:**  Unused. MUST be 0 and ignored.

##### DRS\_MSG\_MOVEREQ\_V2

The DRS\_MSG\_MOVEREQ\_V2 structure defines a request message sent to the [IDL\_DRSInterDomainMove](#Section_595b2fef493b4b1db60de7a1a3345b0e) method.

1. typedef struct {
2. DSNAME\* pSrcDSA;
3. ENTINF\* pSrcObject;
4. DSNAME\* pDstName;
5. DSNAME\* pExpectedTargetNC;
6. DRS\_SecBufferDesc\* pClientCreds;
7. SCHEMA\_PREFIX\_TABLE PrefixTable;
8. ULONG ulFlags;
9. } DRS\_MSG\_MOVEREQ\_V2;

**pSrcDSA:**  The client [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

**pSrcObject:**  The object to be moved.

**pDstName:**  The name the object will have in the destination [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

**pExpectedTargetNC:**  The [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) to which **pSrcObject** is being moved.

**pClientCreds:**  The credentials of the user initiating the call.

**PrefixTable:**  The [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) with which to translate the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values in **pSrcObject** to [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

**ulFlags:**  Unused. MUST be 0 and ignored.

##### DRS\_MSG\_MOVEREPLY

The DRS\_MSG\_MOVEREPLY union defines the response messages received from the [IDL\_DRSInterDomainMove](#Section_595b2fef493b4b1db60de7a1a3345b0e) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_MOVEREPLY\_V1 V1;
6. [case(2)]
7. DRS\_MSG\_MOVEREPLY\_V2 V2;
8. } DRS\_MSG\_MOVEREPLY;

**V1:**  The version 1 response (obsolete).

**V2:**  The version 2 response.

##### DRS\_MSG\_MOVEREPLY\_V1

The DRS\_MSG\_MOVEREPLY\_V1 structure defines a response message received from the [IDL\_DRSInterDomainMove](#Section_595b2fef493b4b1db60de7a1a3345b0e) method. This response version is obsolete.[<35>](#Appendix_A_35" \o "Product behavior note 35)

1. typedef struct {
2. ENTINF\*\* ppResult;
3. SCHEMA\_PREFIX\_TABLE PrefixTable;
4. ULONG\* pError;
5. } DRS\_MSG\_MOVEREPLY\_V1;

**ppResult:**  The [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) as it appears following the move operation.

**PrefixTable:**  The [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) with which to translate the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values in ppResult to [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

**pError:**  0 if successful, or non-zero if a fatal error occurred.

##### DRS\_MSG\_MOVEREPLY\_V2

The DRS\_MSG\_MOVEREPLY\_V2 structure defines a response message received from the [IDL\_DRSInterDomainMove](#Section_595b2fef493b4b1db60de7a1a3345b0e) method.

1. typedef struct {
2. ULONG win32Error;
3. [unique] DSNAME\* pAddedName;
4. } DRS\_MSG\_MOVEREPLY\_V2;

**win32Error:**  0 if successful, or non-zero if a fatal error occurred.

**pAddedName:**  The name of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in its new [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

#### Method-Specific Abstract Types and Procedures

##### AttrIsBacklink

1. procedure AttrIsBacklink(attr: ATTRTYP): boolean

Returns true if the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) *attr* is a back link, and returns false otherwise.

1. return SchemaObj(attr)!linkID mod 2 = 1

##### AttrIsConstructed

1. procedure AttrIsConstructed(attr: ATTRTYP): boolean

Returns true if the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) *attr* is a [**constructed attribute**](#gt_d848b035-c151-4fd8-88d9-9f152d053fee), and returns false otherwise.

1. return FLAG\_ATTR\_IS\_CONSTRUCTED in SchemaObj(attr)!systemFlags

##### AttrIsNonReplicated

1. procedure AttrIsNonReplicated(attr: ATTRTYP): boolean

Returns true if the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) *attr* is a [**nonreplicated attribute**](#gt_6c9b51bd-519b-4f20-97ae-baaf9675f2d7), and returns false otherwise.

1. return FLAG\_ATTR\_NOT\_REPLICATED in SchemaObj(attr)!systemFlags

##### AuthorizationInfoFromClientCredentials

1. procedure AuthorizationInfoFromClientCredentials(
2. credBuffer: DRS\_SecBufferDesc,
3. var token: ClientAuthorizationInfo): DWORD

Generates a [ClientAuthorizationInfo](#Section_92e47a548f244182b6b484f28699f8a1) *token* (which is a security token) from client credentials *credBuffer*. See [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.5.3 for more details. Returns 0 if it succeeds, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if it fails.

##### ImpersonateAuthorizationInfo

1. procedure ImpersonateAuthorizationInfo(token: ClientAuthorizationInfo)

Impersonates a set of client credentials. This affects the outcome of all subsequent [AccessCheckAttr](#Section_48da42bfffae4667937bd5d5627b3ea0), [AccessCheckCAR](#Section_4e482e032f234ee6b3b15cb367013a5d), [AccessCheckObject](#Section_4d2e837695a24bbfaaa9ac26c0257e4e), [AccessCheckWriteToSpnAttribute](#Section_f949fd9a65cf428ca9d6ffa30c2876b4), and related calls, until [RevertToSelf](#Section_675b61f4cad44fac8276139a83d933d4) is called.

##### IsApplicationNC

1. procedure IsApplicationNC(nc: DSName): boolean

Returns true if and only if *nc* is an [**application NC**](#gt_53fee475-b07f-45e1-b4b7-c7ac0c1e7f6a).

##### RevertToSelf

1. procedure RevertToSelf()

Undoes the effect of [ImpersonateAuthorizationInfo](#Section_cd64e651402e4199882d710ce4366c14). After the RevertToSelf procedure is called, the [**security context**](#gt_88d49f20-6c95-4b64-a52c-c3eca2fe5709) is restored to what it was before ImpersonateAuthorizationInfo was called.

#### Server Behavior of the IDL\_DRSInterDomainMove Method

*Informative summary of behavior*: [IDL\_DRSInterDomainMove](#Section_595b2fef493b4b1db60de7a1a3345b0e) is used during a cross-[**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) move operation. This is a special [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) move operation because it involves moving an object from one [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) into another. A normal move operation moves the object within one NC on one DC; a cross-NC move involves two DCs. IDL\_DRSInterDomainMove is an intermediate step in the cross-NC move operation, which is initiated by an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) call. The IDL\_DRSInterDomainMove call is done by the "source" DC to the "target" DC in order to move the object with all of its data from one [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) into another.

**Note**  IDL\_DRSInterDomainMove transfers data that is normally not readable by the end user (such as password hashes and other secrets).

During the move, the [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b) structure that contains the object data is constructed by the source DC and passed to the target DC. The target DC enforces certain constraints, transforms the data according to the processing rules, and then either creates the object in its NC replica or [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the existing object. For more information on cross-NC move operations, see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.5.4.2.

1. ULONG
2. IDL\_DRSInterDomainMove(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_MOVEREQ \*pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DRS\_MSG\_MOVEREPLY \*pmsgOut)
10. msgIn: DRS\_MSG\_MOVEREQ\_V2
11. lastPrefixTableEntry: PrefixTableEntry
12. prefixTable: PrefixTable
13. dwErr: DWORD
14. clientCreds: ClientAuthorizationInfo
15. callerCreds: ClientAuthorizationInfo
16. O: ENTINF
17. existingObj: DSName
18. attribute: ATTRTYP
19. proxyEpoch: DWORD
20. ValidateDRSInput(hDrs, 10)
21. pdwOutVersion^ := 2
22. msgOut^.V2.win32error := ERROR\_DS\_GENERIC\_ERROR
23. msgOut^.V2.pAddedName := null
24. if dwInVersion ≠ 2 then
25. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
26. endif
27. msgIn := pmsgIn^.V2
28. if msgIn.pExpectedTargetNC ≠ DefaultNC() then
29. return ERROR\_DS\_DST\_NC\_MISMATCH
30. endif
31. if msgIn.PrefixTable.PrefixCount < 1 then
32. return ERROR\_SCHEMA\_MISMATCH
33. endif
34. /\* Remember last prefix table entry, and remove it from the prefix
35. \* table.\*/
36. lastPrefixTableEntry :=
37. msgIn.PrefixTable.pPrefixEntry[msgIn.PrefixTable.PrefixCount]
38. msgIn.PrefixTable.PrefixCount := msgIn.PrefixTable.PrefixCount-1
39. /\* Perform a binary comparison of the value from the last
40. \* prefixTable entry with the SchemaInfo.\*/
41. if lastPrefixTableEntry.oid ≠ SchemaInfo() then
42. return ERROR\_SCHEMA\_MISMATCH
43. endif
44. prefixTable := AbstractPTFromConcretePT(msgIn.PrefixTable)
45. /\* Convert client creds into ClientAuthorizationInfo format. \*/
46. dwErr := AuthorizationInfoFromClientCredentials(msgIn.pClientCreds,
47. clientCreds)
48. if dwErr ≠ ERROR\_SUCCESS then
49. return dwErr
50. endif
51. /\* Check that the caller (the "source" DC) is actually a DC by
52. \* checking Enterprise Domain Controllers SID in its token. \*/
53. callerCreds := GetCallerAuthorizationInfo()
54. if not CheckGroupMembership(callerCreds, SidFromStringSid("S-1-5-9"))
55. then
56. return ERROR\_DS\_DRA\_ACCESS\_DENIED
57. endif
58. /\* Validate input ENTINF. \*/
59. O := msgIn.pSrcObject^
60. if ADS\_UF\_SERVER\_TRUST\_ACCOUNT in
61. ENTINF\_GetValue(
62. O, userAccountControl, prefixTable) or
63. ADS\_UF\_INTERDOMAIN\_TRUST\_ACCOUNT in
64. ENTINF\_GetValue(
65. O, userAccountControl, prefixTable) then
66. /\* Disallowed to move DC accounts and trust objects. \*/
67. return ERROR\_DS\_ILLEGAL\_XDOM\_MOVE\_OPERATION
68. endif
69. existingObj := select one obj from all where
70. (obj!distinguishedName = ENTINF\_GetValue(O, distinguishedName,
71. prefixTable))
72. if existingObj ≠ null and
73. existingObj!objectGUID ≠
74. ENTINF\_GetValue(O, objectGUID, prefixTable) then
75. /\* There's already an object with the same DN but different GUID.\*/
76. return ERROR\_DS\_SRC\_GUID\_MISMATCH
77. endif
78. existingObj := select one obj from all where
79. (obj!objectGUID = ENTINF\_GetValue(O, objectGUID, prefixTable))
80. if existingObj ≠ null and existingObj!proxiedObjectName ≠
81. ENTINF\_GetValue(O, proxiedObjectName, prefixTable) then
82. /\* There's already an object with the same guid,
83. \* but proxiedObjectName is different - not allowed. \*/
84. return ERROR\_DS\_EPOCH\_MISMATCH
85. endif
86. if IsApplicationNC(GetObjectNC(O.pName^)) then
87. return ERROR\_DS\_INTERNAL\_FAILURE
88. endif
89. /\* Scan through the ENTINF and throw away any attributes that are not
90. \* supposed to be moved. \*/
91. foreach attribute in ENTINF\_EnumerateAttributes(O, prefixTable)
92. if AttrIsBacklink(attribute) or AttrIsNonReplicated(attribute) or
93. AttrIsConstructed(attribute) then
94. ENTINF\_SetValue(O, attribute, null, prefixTable)
95. endif
96. if attribute in {
97. adminCount, badPasswordTime, badPwdCount, creationTime,
98. distinguishedName, domainReplica, instanceType,
99. isCriticalSystemObject, isDeleted, lastLogoff, lastLogon,
100. lastLogonTimestamp, lockoutTime, logonCount, modifiedCount,
101. modifiedCountAtLastProm, msDS-Cached-Membership,
102. msDS-Cached-Membership-Time-Stamp, msDS-Site-Affinity, nextRid,
103. nTSecurityDescriptor, objectCategory, operatorCount,
104. primaryGroupID, proxiedObjectName, replPropertyMetaData,
105. revision, rid, sAMAccountType, serverState, subRefs,
106. systemFlags, uASCompat, uSNChanged, uSNCreated,
107. uSNDSALastObjRemoved, uSNLastObjRem, whenChanged, whenCreated}
108. then
109. ENTINF\_SetValue(O, attribute, null, prefixTable)
110. endif
111. endfor
112. if ENTINF\_GetValue(O, userAccountControl, prefixTable) ≠ null then
113. /\* Reset lockout bit. \*/
114. ENTINF\_SetValue(O, userAccountControl,
115. ENTINF\_GetValue(O, userAccountControl) - {ADS\_UF\_LOCKOUT},
116. prefixTable)
117. endif
118. if ENTINF\_GetValue(O, pwdLastSet, prefixTable) ≠ null and
119. ENTINF\_GetValue(O, pwdLastSet, prefixTable) ≠ 0 then
120. /\* If pwdLastSet is set to non-zero, then change it to -1. \*/
121. ENTINF\_SetValue(O, pwdLastSet, (LONGLONG)-1, prefixTable)
122. endif
123. /\* Append objectSid to sIDHistory. \*/
124. ENTINF\_SetValue(O, sidHistory,
125. ENTINF\_GetValue(O, sidHistory, prefixTable)
126. + {ENTINF\_GetValue(O, objectSid, prefixTable)})
127. /\* Compute the new proxiedObjectName value. \*/
128. if ENTINF\_GetValue(O, proxiedObjectName, prefixTable) ≠ null and
129. GetProxyType(ENTINF\_GetValue(O, proxiedObjectName)) =
130. PROXY\_TYPE\_MOVED\_OBJECT then
131. /\* There's already a valid proxiedObjectName on the object,
132. \* so just increment the epoch value. \*/
133. proxyEpoch := GetProxyEpoch(ENTINF\_GetValue(O, proxiedObjectName,
134. prefixTable))+1
135. else
136. /\* No valid proxiedObjectName, so start a new one. \*/
137. proxyEpoch := 1
138. endif
139. /\* Stamp the new proxiedObjectName value into ENTINF. \*/
140. ENTINF\_SetValue(O,
141. proxiedObjectName,
142. MakeProxyValue(msgIn.pSrcNC^,
143. PROXY\_TYPE\_MOVED\_OBJECT,
144. proxyEpoch),
145. prefixTable)
146. if existingObj ≠ null then
147. /\* Purge existing object, it is about to be overwritten. \*/
148. Expunge(existingObj)
149. endif
150. ImpersonateAuthorizationInfo(clientCreds)
151. O.pName := msgIn.pDstName
152. dwErr :=
153. PerformAddOperation(
154. O,
155. msgOut^.V2.pAddedName^,
156. AbstractPTFromConcretePT(msgIn.PrefixTable),
157. TRUE)
158. RevertToSelf()
159. msgOut^.V2.win32error := dwErr
160. return dwErr

#### Examples of the IDL\_DRSInterDomainMove Method

In this example, a user is moved from the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) ASIA.CONTOSO.COM to the domain NC CONTOSO.COM.

##### Initial State

Querying the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) for "Aaron Con" in the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) ASIA.CONTOSO.COM on DCA1, prior to the move:

* ldap\_search\_s("CN=Aaron Con,CN=Users,DC=asia,DC=contoso,DC=com", *singleLevel*, "(objectclass=\*)", [*objectClass, cn, ... objectCategory*])
* Getting 1 entries:
* >> Dn: CN=Aaron Con,CN=Users,DC=asia,DC=contoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> cn: Aaron Con;
  + 1> sn: Con;
  + 1> givenName: Aaron;
  + 1> distinguishedName: CN=Aaron Con, CN=Users, DC=asia, DC=contoso, DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/12/2006 17:25:53 Pacific Std Daylight Time;
  + 1> whenChanged: 07/12/2006 17:25:54 Pacific Std Daylight Time;
  + 1> displayName: Aaron Con;
  + 1> uSNCreated: 13798;
  + 1> uSNChanged: 13803;
  + 1> name: Aaron Con;
  + 1> objectGUID: 45a6999f-31eb-40ab-a2e5-906ccd86d5eb;
  + 1> userAccountControl: 0x200 = ( UF\_NORMAL\_ACCOUNT );
  + 1> badPwdCount: 0;
  + 1> codePage: 0;
  + 1> countryCode: 0;
  + 1> badPasswordTime: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogoff: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogon: 01/01/1601 00:00:00 UNC ;
  + 1> pwdLastSet: 07/12/2006 17:25:53 Pacific Std Pacific Daylight Time;
  + 1> primaryGroupID: 513;
  + 1> objectSid: S-1-5-21-1880045291-2375173688-894673254-1109;
  + 1> accountExpires: 09/14/30828 02:48:05 UNC ;
  + 1> logonCount: 0;
  + 1> sAMAccountName: aaroncon;
  + 1> sAMAccountType: 805306368;
  + 1> userPrincipalName: aaroncon@asia.contoso.com;
  + 1> objectCategory: CN=Person, CN=Schema, CN=Configuration, DC=contoso, DC=com;

Querying the user object for "Aaron Con" in the domain NC CONTOSO.COM on DC1 prior to the move yields no results, as follows:

* ldap\_search\_s("CN=Aaron Con,CN=Users,DC=contoso,DC=com", *oneLevel*, "(objectclass=\*)", [*objectClass, cn, ... objectCategory*])
* Error: Search: No Such Object.
* Matched DNs: CN=Users,DC=contoso,DC=com
* Getting 0 entries:

##### Client Request

An [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) client invokes the [IDL\_DRSInterDomainMove](#Section_595b2fef493b4b1db60de7a1a3345b0e) method against a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) named DCA1.ASIA.CONTOSO.COM with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DCA1 omitted):

* *dwInVersion* = 2
* *pmsgIn* = DRS\_MSG\_MOVEREQ\_V2
  + pSrcDSA: CN=NTDS Settings,CN=DCA1,CN=Servers, CN=Default-First-Site-Name,CN=Sites, CN=Configuration,DC=contoso,DC=com
  + pSrcObject: ENTINF
    - objectClass: top; person; organizationalPerson; user
    - cn: Aaron Con
    - sn: Con
    - givenName: Aaron
    - instanceType: IT\_WRITE
    - whenCreated: 07/12/2006 17:25:53 Pacific Std Time Pacific Daylight Time;
    - whenChanged: 07/12/2006 17:25:54 Pacific Std Time Pacific Daylight Time;
    - displayName: Aaron Con;
    - uSNCreated: 13798;
    - uSNChanged: 13803;
    - objectGUID: 45a6999f-31eb-40ab-a2e5-906ccd86d5eb;
    - userAccountControl: 0x200 = ( UF\_NORMAL\_ACCOUNT );
    - badPwdCount: 0;
    - countryCode: 0;
    - badPasswordTime: 01/01/1601 00:00:00 UNC ;
    - lastLogoff: 01/01/1601 00:00:00 UNC ;
    - lastLogon: 01/01/1601 00:00:00 UNC ;
    - dBCSPwd: *Binary data*
    - unicodePwd: *Binary data*
    - supplementalCredentials: *none*
    - pwdLastSet: 07/12/2006 17:25:53 Pacific Std Time Pacific Daylight Time;
    - primaryGroupID: 513;
    - objectSid: S-1-5-21-1880045291-2375173688-894673254-1109;
    - accountExpires: 09/14/30828 02:48:05 UNC ;
    - logonCount: 0;
    - sAMAccountName: aaroncon;
    - sAMAccountType: SAM\_NORMAL\_USER\_ACCOUNT;
      * userPrincipalName: aaroncon@asia.contoso.com;
    - objectCategory: CN=Person, CN=Schema, CN=Configuration, DC=contoso, DC=com;
    - nTSecurityDescriptor: *Binary data*
    - replPropertyMetaData: *omitted*
* pDstName: CN=Aaron Con,CN=Users,DC=contoso,DC=com
* pExpectedTargetNC: DC=contoso,DC=com
* pClientCreds: [DRS\_SecBufferDesc](#Section_aa6f5b36cf1e49c9b5fd4cd5c9de7448)
* PrefixTable: [SCHEMA\_PREFIX\_TABLE](#Section_9b371267e8b84c69997902dae02e5e38)
* ulFlags: None

##### Server Response

Return code of 0 with the following values:

* *pdwOutVersion* = 2
* *pmsgOut* = [DRS\_MSG\_MOVEREPLY\_V2](#Section_59b23876f18c44bdb88f3848d256c61e)
  + pAddedName: CN=Aaron Con,CN=Users,DC=contoso,DC=com

##### Final State

After the move, the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) for "Aaron Con" is not present on [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) ASIA.CONTOSO.COM, querying DCA1, as follows:

* ldap\_search\_s("CN=Aaron Con,CN=Users,DC=asia,DC=contoso,DC=com", *singleLevel*, "(objectclass=\*)", [*distinguishedName, objectGUID, userAccountControl, objectSid, sAMAccountName, userPrincipalName*])
* Error: Search: No Such Object.
* Matched DNs: CN=Users,DC=asia,DC=contoso,DC=com
* Getting 0 entries:

After the move, the user object for "Aaron Con" is now present on domain NC CONTOSO.COM, querying DC1:

* ldap\_search\_s("CN=Aaron Con,CN=Users, DC=contoso,DC=com", *oneLevel*, "(objectclass=\*)", [*cn, distinguishedName ... proxiedObjectName, dSCorePropagationData*])
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=Aaron Con,CN=Users,DC=contoso,DC=com
  + 4> objectClass: top; person; organizationalPerson; user;
  + 1> cn: Aaron Con;
  + 1> sn: Con;
  + 1> givenName: Aaron;
  + 1> distinguishedName: CN=Aaron Con,CN=Users,DC=contoso,DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/12/2006 17:32:04 Pacific Standard Daylight Time;
  + 1> whenChanged: 07/12/2006 17:32:04 Pacific Standard Daylight Time;
  + 1> displayName: Aaron Con;
  + 1> uSNCreated: 15366;
  + 1> uSNChanged: 15369;
  + 1> name: Aaron Con;
  + 1> objectGUID: 45a6999f-31eb-40ab-a2e5-906ccd86d5eb;
  + 1> userAccountControl: 0x200 = ( UF\_NORMAL\_ACCOUNT );
  + 1> badPwdCount: 0;
  + 1> codePage: 0;
  + 1> countryCode: 0;
  + 1> badPasswordTime: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogoff: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogon: 01/01/1601 00:00:00 UNC ;
  + 1> pwdLastSet: 07/12/2006 17:32:04 Pacific Standard Daylight Time;
  + 1> primaryGroupID: 513;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1111;
  + 1> accountExpires: 09/14/30828 02:48:05 UNC ;
  + 1> logonCount: 0;
  + 1> sAMAccountName: aaroncon;
  + 1> sAMAccountType: 805306368;
  + 1> sIDHistory: S-1-5-21-1880045291-2375173688-894673254-1109;
  + 1> userPrincipalName: aaroncon@asia.contoso.com;
  + 1> objectCategory: CN=Person, CN=Schema, CN=Configuration, DC=contoso, DC=com;
  + 1> proxiedObjectName: B:16:0000000000000001:DC=asia, DC=contoso, DC=com;
  + 1> dSCorePropagationData: 07/12/2006 17:32:04 Pacific Standard Daylight Time; 07/12/2006 17:32:04 Pacific Standard Time Pacific Daylight Time; 01/01/1601 01:08:16 UNC ;

### IDL\_DRSQuerySitesByCost (Opnum 24)

The IDL\_DRSQuerySitesByCost method determines the communication cost from a "from" [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba) to one or more "to" sites.

1. ULONG IDL\_DRSQuerySitesByCost(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_QUERYSITESREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_QUERYSITESREPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if a failure occurs.

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_QUERYSITESREQ

The DRS\_MSG\_QUERYSITESREQ union defines the request message versions sent to the [IDL\_DRSQuerySitesByCost](#Section_2c3faba2d64e4866b8f1fc8d5f4ec710) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_QUERYSITESREQ\_V1 V1;
6. } DRS\_MSG\_QUERYSITESREQ;

**V1:**  The version 1 request.

##### DRS\_MSG\_QUERYSITESREQ\_V1

The DRS\_MSG\_QUERYSITESREQ\_V1 structure defines a request message sent to the [IDL\_DRSQuerySitesByCost](#Section_2c3faba2d64e4866b8f1fc8d5f4ec710) method.

1. typedef struct {
2. [string] const WCHAR\* pwszFromSite;
3. [range(1,10000)] DWORD cToSites;
4. [string, size\_is(cToSites)] WCHAR\*\* rgszToSites;
5. DWORD dwFlags;
6. } DRS\_MSG\_QUERYSITESREQ\_V1;

**pwszFromSite:**  The [**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) of the site [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of the "from" [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba).

**cToSites:**  The number of items in the **rgszToSites** array (the count of "to" sites).

**rgszToSites:**  The RDNs of the site objects of the "to" sites.

**dwFlags:**  Unused. MUST be 0 and ignored.

##### DRS\_MSG\_QUERYSITESREPLY

The DRS\_MSG\_QUERYSITESREPLY union defines the response messages received from the [IDL\_DRSQuerySitesByCost](#Section_2c3faba2d64e4866b8f1fc8d5f4ec710) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_QUERYSITESREPLY\_V1 V1;
6. } DRS\_MSG\_QUERYSITESREPLY;

**V1:**  The version 1 response.

##### DRS\_MSG\_QUERYSITESREPLY\_V1

The DRS\_MSG\_QUERYSITESREPLY\_V1 structure defines a response message received from the [IDL\_DRSQuerySitesByCost](#Section_2c3faba2d64e4866b8f1fc8d5f4ec710) method.

1. typedef struct {
2. [range(0,10000)] DWORD cToSites;
3. [size\_is(cToSites)] DRS\_MSG\_QUERYSITESREPLYELEMENT\_V1\* rgCostInfo;
4. DWORD dwFlags;
5. } DRS\_MSG\_QUERYSITESREPLY\_V1;

**cToSites:**  The number of items in the **rgCostInfo** array.

**rgCostInfo:**  The sequence of computed [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba) costs, in the same order as the **rgszToSites** field in the request message.

**dwFlags:**  Unused. MUST be 0 and ignored.

##### DRS\_MSG\_QUERYSITESREPLYELEMENT\_V1

The DRS\_MSG\_QUERYSITESREPLYELEMENT\_V1 structure defines the computed cost of communication between two [**sites**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba).

1. typedef struct {
2. DWORD dwErrorCode;
3. DWORD dwCost;
4. } DRS\_MSG\_QUERYSITESREPLYELEMENT\_V1;

**dwErrorCode:**  0 if this "from-to" computation was successful, or ERROR\_DS\_OBJ\_NOT\_FOUND if the "to" site does not exist.

**dwCost:**  The communication cost between the "from" site and this "to" site, or 0xFFFFFFFF if the sites are not connected.

#### Method-Specific Abstract Types and Procedures

##### ValidateSiteRDN

1. procedure ValidateSiteRDN(s: unicodestring): boolean

*Informative summary of behavior*: The ValidateSiteRDN procedure returns 0 if *s* is a valid [**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) for a site [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca), and returns an appropriate error otherwise. A valid RDN has the following characteristics:

* Is not null.
* Does not have 0 length.
* Does not have a length greater than 64.
* Contains no occurrences of the equal sign (=) or comma (,).

1. if s = null or s.length = 0 then
2. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
3. endif
4. if s.Length > 64 then
5. return ERROR\_DS\_NAME\_TOO\_LONG
6. endif
7. if s contains (=) or s contains (,) then
8. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
9. endif
10. return 0

##### WeightedArc and WeightedArcSet

1. type WeightedArc = [initial: DSName, final: DSName, cost: integer]
2. type WeightedArcSet = set of WeightedArc

The cost field of a WeightedArc is positive.

##### MinWeightPath

1. procedure MinWeightPath(
2. vSet: set of DSName,
3. aSet: WeightedArcSet): WeightedArcSet

Returns a WeightedArcSet where, for each WeightedArc *a*:

* *a.initial* and *a.final* are vertices in *vSet*
* *a.final* is reachable from *a.initia*l in the graph G = (*vSet*, *aSet*)
* *a.cost* is the cost of the minimum-cost path in G from *a.initial* to *a.final*.

#### Server Behavior of the IDL\_DRSQuerySitesByCost Method

*Informative summary of behavior*: Given a [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba) *fromSite* and an array of sites *toSites*, the server returns an array that contains the cost from *fromSite* to each element of *toSite*, where the cost is defined as follows.

The server computes a weighted graph G = (V, A). Each vertex in V corresponds to a site [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). Each arc in A corresponds to a siteLink object that connects two vertices in V; the weight of an arc is the value of [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) cost on the arc's siteLink object. The cost of a path in the graph is the sum of the arc weights on the path. The cost from one site to another is the minimum-cost path between the two sites.

The model just described corresponds to fully transitive communications between sites: If site *a* communicates with site *b* and site *b* communicates with site *c*, then site *a* communicates with site *c* by routing through *b*. Replication can be configured to restrict transitive communication to sites specified in the same siteLinkBridge object. Suppose there is a siteLink object for site *a* and site *b*, and a siteLink object for site *b* and site *c*, but no siteLink object for site *a* and site *c*. If both of the siteLink objects are specified on the same siteLinkBridge object, site *a* can communicate with site *c* by routing through *b*. If no such siteLinkBridge object exists, site *a* cannot communicate with site *c*.

To calculate the cost when siteLinkBridge objects are used, let *nBridges* be the number of siteLinkBridge objects. For each k in the subrange [0 .. *nBridges*-1], construct a weighted graph G[k] = (V, A[k]) using siteLinkBridge object b[k]. Graph G[k] has the same vertex set as G, but its arc set A[k] is a subset of A, including only the arcs listed in attribute siteLinkList on siteLinkBridge object b[k]. Then the cost from site *a* to site *c* is the minimum of the following costs:

1. The cost of the arc, if any, from *a* to *c* in G.
2. For each k in the subrange [0 .. *nBridges*-1], the cost of the minimum cost path, if any, from *a* to *c* in G[k].

Any authenticated user can perform this operation; no access checking is performed.[<36>](#Appendix_A_36" \o "Product behavior note 36)

1. ULONG
2. IDL\_DRSQuerySitesByCost(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_QUERYSITESREQ \*pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DRS\_MSG\_QUERYSITESREPLY \*pmsgOut)
10. msgIn: DRS\_MSG\_QUERYSITESREQ\_V1
11. vSet, slSet, sbSet : set of DSName
12. aSet, aSetB, aSetC, aSetD: WeightedArcSet
13. siteContainer, ipObject, fromSite, toSite: DSName
14. u, v, sl, sb: DSName
15. i, c: integer
16. min: WeightedArc
17. ul : ULONG
18. ValidateDRSInput(hDrs, 24)
19. pdwOutVersion^ := 1
20. pmsgOut^.V1.cToSites := 0
21. pmsgOut^.V1. rgCostInfo := null
22. pmsgOut^.V1.dwFlags := 0
23. /\* Perform input validation,
24. \* initialize siteContainer, ipObject, fromSite. \*/
25. if dwInVersion ≠ 1 then
26. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
27. endif
28. msgIn := pmsgIn^.V1
29. ul := ValidateSiteRDN(msgIn.pwszFromSite)
30. if 0 ≠ ul then
31. return ul
32. endif
33. if msgIn.cToSites > 0 and msgIn.rgszToSites = null then
34. return ERROR\_INVALID\_PARAMETER
35. endif
36. for i := 0 to msgIn.cToSites – 1
37. ul := ValidateSiteRDN(msgIn.rgszToSites[i])
38. if 0 ≠ ul then
39. return ul
40. endif
41. endfor
42. siteContainer := DescendantObject(ConfigNC(), "CN=Sites,")
43. ipObject := DescendantObject(ConfigNC(),
44. "CN=IP,CN=Inter-Site Transports,CN=Sites,")
45. fromSite := select one v from children siteContainer where
46. site in v!objectClass and v!name = msgIn.pwszFromSite
47. if fromSite = null then
48. return ERROR\_DS\_OBJ\_NOT\_FOUND
49. endif
50. /\* Construct the vertex set vSet. \*/
51. vSet := select all v from children siteContainer where
52. site in v!objectClass
53. if vSet = {} then
54. return ERROR\_DS\_OBJ\_NOT\_FOUND
55. endif
56. /\* Construct the arc set aSet. \*/
57. slSet := select all v from children ipObject where
58. siteLink in v!objectClass
59. foreach sl in slSet
60. foreach u in sl!siteList
61. foreach v in sl!siteList - {u}
62. aSet := aSet + {[initial: u, final: v, cost: sl!cost]}
63. endfor
64. endfor
65. endfor
66. /\* Construct minimum-cost arc set aSetC.
67. \* See [MS-ADTS] section 6.1.1.2.2.3.1, "IP Transport Container", for
68. \* the definition of the NTDSTRANSPORT\_OPT\_BRIDGES\_REQUIRED option. \*/
69. if NTDSTRANSPORT\_OPT\_BRIDGES\_REQUIRED in ipObject!options then
70. /\* Perform construction using siteLinkBridge objects.
71. \* Initial minimum cost is the cost of a direct arc if any. \*/
72. aSetC := aSet
73. sbSet := select all v from children ipObject where
74. siteLinkBridge in v!objectclass
75. foreach sb in sbSet
76. /\* Compute the minimum cost using this siteLinkBridge. \*/
77. aSetB := {}
78. foreach sl in sb!siteLinkList
79. foreach u in sl!siteList
80. foreach v in sl!siteList - {u}
81. aSetB := aSetB + {[initial: u, final: v, cost: sl!cost)}
82. endfor
83. endfor
84. endfor
85. aSetD := MinWeightPath(vSet, aSetB)
86. /\* Here aSetD contains the minimum cost arc set using this
87. \* siteLinkBridge. Improve the current minimum cost using
88. \* aSetD. \*/
89. foreach [initial: u, final: v, cost: c] in aSetD
90. min := select one t from aSetC where
91. t.initial = u and t.final = v
92. if min = null then
93. aSetC := aSetC + {[initial: u, final: v, cost: c)}
94. else if min.cost > c then
95. aSetC := aSetC - {[initial: u, final: v, cost: min.cost]}
96. + {[initial: u, final: v, cost: c)}
97. endif
98. endfor
99. endfor
100. else
101. /\* Fully transitive network, ignore siteLinkBridge objects. \*/
102. aSetC := MinWeightPath(vSet, aSet)
103. endif
104. /\* Construct result message. \*/
105. pdwOutVersion^ := 1
106. pmsgOut^.V1.cToSites := msgIn.cToSites
107. pmsgOut^.V1.dwFlags := 0
108. for i:= 0 to msgIn.cToSites - 1
109. toSite := select one v from children siteContainer where
110. site in v!objectClass and v!name = msgIn.rgszToSites[i]
111. if not (toSite in vSet) then
112. pmsgOut^.V1.rgCostInfo[i].dwErrorCode := ERROR\_DS\_OBJ\_NOT\_FOUND
113. pmsgOut^.V1.rgCostInfo[i].dwCost := 0xffffffff
114. else
115. min := select one t from aSetC where
116. t.initial = fromSite and t.final = toSite
117. if min ≠ null then
118. pmsgOut^.V1.rgCostInfo[i].dwErrorCode := 0
119. pmsgOut^.V1.rgCostInfo[i].dwCost := min.cost
120. else
121. pmsgOut^.V1.rgCostInfo[i].dwErrorCode = 0
122. pmsgOut^.V1.rgCostInfo[i].dwCost := 0xffffffff
123. endif
124. endif
125. endfor
126. return 0

#### Examples of IDL\_DRSQuerySitesByCost Method

##### Nontransitive Communication Using siteLinkBridge

Determines the nontransitive communication cost "from" [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba) DC1 "to" sites DC2, DC3, and DC4. A site graph is displayed in the following figure.

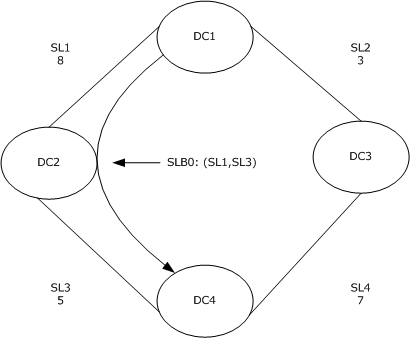


Figure 2: Site graph for a nontransitive network

| VERTEX | ARC | ARC WEIGHT |
| --- | --- | --- |
| DC1 | SL1  SL2  SLB0 (SL1, SL3) | 8  3  13 |
| DC2 | SL1  SL3 | 8  5 |
| DC3 | SL2  SL4 | 3  7 |
| DC4 | SL3  SL4  SLB0 (SL1, SL3) | 7  13 |

###### Initial State

Querying the [**site object**](#gt_0ce6abc5-9823-4a69-bb30-12e42ff99629) for [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) CONTOSO.COM by performing an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) search with Base [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) "CN=Configuration,DC=contoso,DC=com".

* ldap\_search\_s(ld, "CN=Configuration,DC=contoso,DC=com", 2, "(objectclass=site)", attrList, 0, &msg)
* Result <0>: (null)
* Matched DNs:
* Getting 4 entries:
* >> Dn: CN=DC1,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + 1> cn: DC1;
  + 1> distinguishedName: CN=DC1,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> instanceType: 0x4 = (WRITE);
  + 1> location: DC1;
  + 1> name: DC1;
  + 1> objectCategory: CN=Site,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; site;
  + 1> objectGUID: 3540d101-be2d-4630-b75e-1343c2a39dc8;
  + 1> showInAdvancedViewOnly: TRUE;
  + 1> systemFlags: 0x42000000 = (CONFIG\_ALLOW\_RENAME | DISALLOW\_MOVE\_ON\_DELETE);
  + 1> uSNChanged: 41007;
  + 1> uSNCreated: 36885;
  + 1> whenChanged: 06/08/2010 19:04:05 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 13:53:19 Pacific Standard Time;
* >> Dn: CN=DC2,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + 1> cn: DC2;
  + 1> distinguishedName: CN=DC2,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = (WRITE);
  + 1> location: DC2;
  + 1> name: DC2
  + 1> objectCategory: CN=Site,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; site;
  + 1> objectGUID: 7dd0525e-f00a-4c1d-9eec-d6df02625a59;
  + 1> showInAdvancedViewOnly: TRUE;
  + 1> systemFlags: 0x42000000 = (CONFIG\_ALLOW\_RENAME | DISALLOW\_MOVE\_ON\_DELETE);
  + 1> uSNChanged: 40991;
  + 1> uSNCreated: 40991
  + 1> whenChanged: 06/08/2010 18:39:43 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 18:39:43 Pacific Standard Time;
* >> Dn: CN=DC3,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + 1> cn: DC3;
  + 1> distinguishedName: CN=DC3,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = (WRITE);
  + 1> location: DC3;
  + 1> name: DC3;
  + 1> objectCategory: CN=Site,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; site;
  + 1> objectGUID: dbdff472-a414-44c2-8206-a619e5eee583;
  + 1> showInAdvancedViewOnly: TRUE;
  + 1> systemFlags: 0x42000000 = (CONFIG\_ALLOW\_RENAME | DISALLOW\_MOVE\_ON\_DELETE);
  + 1> uSNChanged: 40997;
  + 1> uSNCreated: 40997;
  + 1> whenChanged: 06/08/2010 18:53:31 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 18:53:31 Pacific Standard Time;
* >> Dn: CN=DC4,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + 1> cn: DC4;
  + 1> distinguishedName: CN=DC4,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = (WRITE);
  + 1> location: DC4;
  + 1> name: DC4;
  + 1> objectCategory: CN=Site,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; site;
  + 1> objectGUID: c15325f5-881b-417a-80cf-8e3530885613;
  + 1> showInAdvancedViewOnly: TRUE;
  + 1> systemFlags: 0x42000000 = (CONFIG\_ALLOW\_RENAME | DISALLOW\_MOVE\_ON\_DELETE);
  + 1> uSNChanged: 41002;
  + 1> uSNCreated: 41002;
  + 1> whenChanged: 06/08/2010 18:59:28 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 18:59:28 Pacific Standard Time;

Querying the siteLink [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for domain NC CONTOSO.COM by performing an LDAP search with Base DN "CN=Configuration,DC=contoso,DC=com".

* ldap\_search\_s(ld, "CN=Configuration,DC=contoso,DC=com", 2, "(objectclass=sitelink)", attrList, 0, &msg)
* Result <0>: (null)
* Matched DNs:
* Getting 4 entries:
* >> Dn: CN=SL1, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com
  + 1> cn: SL1;
  + 1> cost: 8;
  + 1> distinguishedName: CN=SL1, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> instanceType: 0x4 = ( WRITE );
  + 1> name: SL1;
  + 1> objectCategory: CN=Site-Link,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; siteLink;
  + 1> objectGUID: bd4ba671-90fb-4f4b-ab5d-76c9451d300c;
  + 1> replInterval: 180;
  + 2> siteList : CN=DC2,CN=Sites, CN=Configuration, DC=contoso, DC=com; CN=DC1, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> systemFlags: 0x40000000 = ( CONFIG\_ALLOW\_RENAME );
  + 1> uSNChanged: 41010;
  + 1> uSNCreated: 36896;
  + 1> whenChanged: 06/08/2010 19:04:37 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 14:01:17 Pacific Standard Time;
* >> Dn: CN=SL2, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com
  + 1> cn: SL2;
  + 1> cost: 3;
  + 1> distinguishedName: CN=SL2, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = ( WRITE );
  + 1> name: SL2;
  + 1> objectCategory: CN=Site-Link,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; siteLink;
  + 1> objectGUID: a8906f5f-0c46-4276-87c6-34e60c6c0d63;
  + 1> replInterval: 180;
  + 2> siteList: CN=DC3, CN=Sites, CN=Configuration, DC=contoso, DC=com; CN=DC1, CN=Sites, CN=Configuration,DC=contoso,DC=com;
  + 1> systemFlags: 0x40000000 = ( CONFIG\_ALLOW\_RENAME );
  + 1> uSNChanged: 41014;
  + 1> uSNCreated: 41014;
  + 1> whenChanged: 06/08/2010 19:05:29 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 19:05:29 Pacific Standard Time;
* >> Dn: CN=SL3, CN=IP, CN=Intersite Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com
  + 1> cn: SL3;
  + 1> cost: 5;
  + 1> distinguishedName: CN=SL3, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = ( WRITE);
  + 1> name: SL3;
  + 1> objectCategory: CN=Site-Link, CN=Schema, CN=Configuration, DC=contoso, DC=com;
  + 2> objectClass: top; siteLink;
  + 1> objectGUID: 33f2a214-bea7-4061-8ecf-eca598837bc3;
  + 1> replInterval: 180;
  + 2> siteList: CN=DC4,CN=Sites, CN=Configuration, DC=contoso, DC=com; CN=DC2, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> systemFlags: 0x40000000 = ( CONFIG\_ALLOW\_RENAME);
  + 1> uSNChanged: 41017;
  + 1> uSNCreated: 41017;
  + 1> whenChanged: 06/08/2010 19:05:51 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 19:05:51 Pacific Standard Time;
* >> Dn: CN=SL4, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com
  + 1> cn: SL4;
  + 1> cost: 7;
  + 1> distinguishedName: CN=SL4, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = ( WRITE );
  + 1> name: SL4;
  + 1> objectCategory: CN=Site-Link,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; siteLink;
  + 1> objectGUID: 3c3e2aa6-03b3-4aab-a0b2-a689a7636619;
  + 1> replInterval: 180;
  + 2> siteList: CN=DC4, CN=Sites, CN=Configuration, DC=contoso, DC=com; CN=DC3, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> systemFlags: 0x40000000 = ( CONFIG\_ALLOW\_RENAME );
  + 1> uSNChanged: 41020;
  + 1> uSNCreated: 41020;
  + 1> whenChanged: 06/08/2010 19:06:13 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 19:06:13 Pacific Standard Time;

Querying the siteLinkBridge object for domain NC CONTOSO.COM by performing an LDAP search with Base DN "CN=Configuration,DC=contoso,DC=com".

* ldap\_search\_s(ld, "CN=Configuration,DC=contoso,DC=com", 2, "(objectclass=sitelinkbridge)", attrList, 0, &msg)
* Result <0>: (null)
* Matched DNs:
* Getting 1 entry:
* >> Dn: CN=SLB0, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com
  + 1> cn: SLB0;
  + 1> distinguishedName: CN=SLB0, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> dSCorePropagationData (2): 06/08/2010 17:06:09 Pacific Standard Time; 0x1 = (NEW\_SD);
  + 1> instanceType: 0x4 = (WRITE);
  + 1> name: SLB0;
  + 1> objectCategory: CN=Site-Link-Bridge,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; siteLinkBridge;
  + 1> objectGUID: 6ed39e2c-0bb4-4fe7-9cb1-5b4e82d1a5e2;
  + 2> siteLinkList: CN=SL1, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso,DC=com; CN=SL3, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> systemFlags: 0x40000000 = ( CONFIG\_ALLOW\_RENAME );
  + 1> uSNChanged: 36899;
  + 1> uSNCreated: 36899;
  + 1> whenChanged: 06/08/2010 14:05:25 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 14:05:25 Pacific Standard Time;

###### Client Request

A client invokes the [IDL\_DRSQuerySitesByCost](#Section_2c3faba2d64e4866b8f1fc8d5f4ec710) method against Contoso with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 is omitted):

* *dwInVersion* =1
* *pmsgIn* = DRS\_MSG\_QUERYSITESREQ\_V1
  + pwszFromSite = "DC1"
  + cToSites =3
  + rgszToSites = {"DC2", "DC3", "DC4"}
* dwFlags =0

###### Server Response

* *pdwOutVersion^* = 1
* *pmsgOut* = DRS\_MSG\_QUERYSITESREPLY\_V1
  + cToSites =3
  + rgCostInfo[0]: DRS\_MSG\_QUERYSITESREPLYELEMENT\_V1
    - dwErrorCode =0
    - dwCost: =8
  + rgCostInfo[1]: DRS\_MSG\_QUERYSITESREPLYELEMENT\_V
    - dwErrorCode =0
    - dwCost: =3
  + rgCostInfo[2]: DRS\_MSG\_QUERYSITESREPLYELEMENT\_V1
    - dwErrorCode =0
    - dwCost: =13
* dwFlags =0

###### Final State

The final state is the same as the initial state; there is no change.

##### Transitive Communication

Determines the transitive communication cost from [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba) DC1 to sites DC2, DC3, and DC4. A site graph is displayed in the following figure.

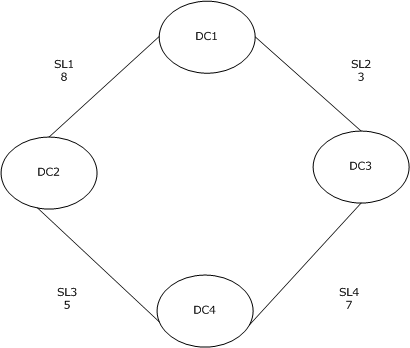


Figure 3: Site graph for a transitive network

| VERTEX | ARC | ARC WEIGHT |
| --- | --- | --- |
| DC | SL1  SL2 | 8  3 |
| DC2 | SL1  SL3 | 8  5 |
| DC3 | SL2  SL4 | 3  7 |
| DC4 | SL3  SL4 | 5  7 |

###### Initial State

Querying the [**site object**](#gt_0ce6abc5-9823-4a69-bb30-12e42ff99629) for [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) CONTOSO.COM by performing an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) search with Base [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) "CN=Configuration,DC=contoso,DC=com".

* ldap\_search\_s(ld, "CN=Configuration,DC=contoso,DC=com", 2, "(objectclass=site)", attrList, 0, &msg)
* Result <0>: (null)
* Matched DNs;
* Getting 4 entries;
* >> Dn: CN=DC1,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + 1> cn: DC1;
  + 1> distinguishedName: CN=DC1,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> instanceType: 0x4 = (WRITE);
  + 1> location: DC1;
  + 1> name: DC1;
  + 1> objectCategory: CN=Site,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; site;
  + 1> objectGUID: 3540d101-be2d-4630-b75e-1343c2a39dc8;
  + 1> showInAdvancedViewOnly: TRUE;
  + 1> systemFlags: 0x42000000 = (CONFIG\_ALLOW\_RENAME | DISALLOW\_MOVE\_ON\_DELETE);
  + 1> uSNChanged: 41007;
  + 1> uSNCreated: 36885;
  + 1> whenChanged: 06/08/2010 19:04:05 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 13:53:19 Pacific Standard Time;
* >> Dn: CN=DC2,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + 1> cn: DC2;
  + 1> distinguishedName: CN=DC2,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = (WRITE);
  + 1> location: DC2;
  + 1> name: DC2;
  + 1> objectCategory: CN=Site,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; site;
  + 1> objectGUID: 7dd0525e-f00a-4c1d-9eec-d6df02625a59;
  + 1> showInAdvancedViewOnly: TRUE;
  + 1> systemFlags: 0x42000000 = (CONFIG\_ALLOW\_RENAME | DISALLOW\_MOVE\_ON\_DELETE);
  + 1> uSNChanged: 40991;
  + 1> uSNCreated: 40991;
  + 1> whenChanged: 06/08/2010 18:39:43 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 18:39:43 Pacific Standard Time;
* >> Dn: CN=DC3,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + 1> cn: DC3;
  + 1> distinguishedName: CN=DC3,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = (WRITE);
  + 1> location: DC3;
  + 1> name: DC3;
  + 1> objectCategory: CN=Site,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; site;
  + 1> objectGUID: dbdff472-a414-44c2-8206-a619e5eee583;
  + 1> showInAdvancedViewOnly: TRUE;
  + 1> systemFlags: 0x42000000 = (CONFIG\_ALLOW\_RENAME | DISALLOW\_MOVE\_ON\_DELETE);
  + 1> uSNChanged: 40997;
  + 1> uSNCreated: 40997;
  + 1> whenChanged: 06/08/2010 18:53:31 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 18:53:31 Pacific Standard Time;
* >> Dn: CN=DC4,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + > cn: DC4;
  + 1> distinguishedName: CN=DC4,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = (WRITE);
  + 1> location: DC4;
  + 1> name: DC4;
  + 1> objectCategory: CN=Site,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; site;
  + 1> objectGUID: c15325f5-881b-417a-80cf-8e3530885613;
  + 1> showInAdvancedViewOnly: TRUE;
  + 1> systemFlags: 0x42000000 = (CONFIG\_ALLOW\_RENAME | DISALLOW\_MOVE\_ON\_DELETE);
  + 1> uSNChanged: 41002;
  + 1> uSNCreated: 41002;
  + 1> whenChanged: 06/08/2010 18:59:28 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 18:59:28 Pacific Standard Time;

Querying the siteLink [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for domain NC CONTOSO.COM by performing an LDAP search with Base DN "CN=Configuration,DC=contoso,DC=com".

* ldap\_search\_s(ld, "CN=Configuration,DC=contoso,DC=com", 2, "(objectclass=sitelink)", attrList, 0, &msg)
* Result <0>: (null)
* Matched DNs;
* Getting 4 entries:
* >> Dn: CN=SL1, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com
  + 1> cn: SL1;
  + 1> cost: 8;
  + 1> distinguishedName: CN=SL1, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> instanceType: 0x4 = ( WRITE );
  + 1> name: SL1;
  + 1> objectCategory: CN=Site-Link,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; siteLink;
  + 1> objectGUID: bd4ba671-90fb-4f4b-ab5d-76c9451d300c;
  + 1> replInterval: 180;
  + 2> siteList : CN=DC2,CN=Sites, CN=Configuration, DC=contoso, DC=com; CN=DC1, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> systemFlags: 0x40000000 = ( CONFIG\_ALLOW\_RENAME );
  + 1> uSNChanged: 41010;
  + 1> uSNCreated: 36896;
  + 1> whenChanged: 06/08/2010 19:04:37 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 14:01:17 Pacific Standard Time;
* >> Dn: CN=SL2, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> cn: SL2;
  + 1> cost: 3;
  + 1> distinguishedName: CN=SL2, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = ( WRITE );
  + 1> name: SL2;
  + 1> objectCategory: CN=Site-Link,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; siteLink;
  + 1> objectGUID: a8906f5f-0c46-4276-87c6-34e60c6c0d63;
  + 1> replInterval: 180;
  + 2> siteList: CN=DC3, CN=Sites, CN=Configuration, DC=contoso, DC=com; CN=DC1, CN=Sites, CN=Configuration,DC=contoso,DC=com;
  + 1> systemFlags: 0x40000000 = ( CONFIG\_ALLOW\_RENAME );
  + 1> uSNChanged: 41014;
  + 1> uSNCreated: 41014;
  + 1> whenChanged: 06/08/2010 19:05:29 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 19:05:29 Pacific Standard Time;
* >> Dn: CN=SL3, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com
  + 1> cn: SL3;
  + 1> cost: 5;
  + 1> distinguishedName: CN=SL3, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = ( WRITE );
  + 1> name: SL3;
  + 1> objectCategory: CN=Site-Link, CN=Schema, CN=Configuration, DC=contoso, DC=com;
  + 2> objectClass: top; siteLink;
  + 1> objectGUID: 33f2a214-bea7-4061-8ecf-eca598837bc3;
  + 1> replInterval: 180;
  + 2> siteList: CN=DC4,CN=Sites, CN=Configuration, DC=contoso, DC=com; CN=DC2, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> systemFlags: 0x40000000 = ( CONFIG\_ALLOW\_RENAME );
  + 1> uSNChanged: 41017;
  + 1> uSNCreated: 41017;
  + 1> whenChanged: 06/08/2010 19:05:51 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 19:05:51 Pacific Standard Time;
* >> Dn: CN=SL4, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com
  + 1> cn: SL4;
  + 1> cost: 7;
  + 1> distinguishedName: CN=SL4, CN=IP, CN=Inter-Site Transports, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = ( WRITE );
  + 1> name: SL4;
  + 1> objectCategory: CN=Site-Link,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; siteLink;
  + 1> objectGUID: 3c3e2aa6-03b3-4aab-a0b2-a689a7636619;
  + 1> replInterval: 180;
  + 2> siteList: CN=DC4, CN=Sites, CN=Configuration, DC=contoso, DC=com; CN=DC3, CN=Sites, CN=Configuration, DC=contoso, DC=com;
  + 1> systemFlags: 0x40000000 = ( CONFIG\_ALLOW\_RENAME );
  + 1> uSNChanged: 41020;
  + 1> uSNCreated: 41020;
  + 1> whenChanged: 06/08/2010 19:06:13 Pacific Standard Time;
  + 1> whenCreated: 06/08/2010 19:06:13 Pacific Standard Time;

###### Client Request

A client invokes the [IDL\_DRSQuerySitesByCost](#Section_2c3faba2d64e4866b8f1fc8d5f4ec710) method against Contoso with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 is omitted):

* *dwInVersion* = 1
* *pmsgIn* = DRS\_MSG\_QUERYSITESREQ\_V1
  + pwszFromSite = "DC1"
  + cToSites =3
  + rgszToSites = {"DC2", "DC3", "DC4"}
* dwFlags =0

###### Server Response

* *pdwOutVersion^* = 1
* *pmsgOut* = DRS\_MSG\_QUERYSITESREPLY\_V1
  + cToSites =3
  + rgCostInfo[0]: DRS\_MSG\_QUERYSITESREPLYELEMENT\_V1
    - dwErrorCode = 0
    - dwCost: = 8
  + rgCostInfo[1]: DRS\_MSG\_QUERYSITESREPLYELEMENT\_V1
    - dwErrorCode = 0
    - dwCost: = 3
  + rgCostInfo[2]: DRS\_MSG\_QUERYSITESREPLYELEMENT\_V1
    - dwErrorCode = 0
    - dwCost: = 10
* dwFlags = 0

###### Final State

The final state is the same as the initial state; there is no change.

### IDL\_DRSRemoveDsDomain (Opnum 15)

The IDL\_DRSRemoveDsDomain method removes the representation (also known as metadata) of a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) from the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9).

1. ULONG IDL\_DRSRemoveDsDomain(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_RMDMNREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_RMDMNREPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message. This must be set to 1, because this is the only version supported.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message. The value must be 1 because that is the only version supported.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if a failure occurs.

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_RMDMNREQ

The DRS\_MSG\_RMDMNREQ union defines the request messages sent to the [IDL\_DRSRemoveDsDomain](#Section_aa3cfa46c737425aae65ecaf9efe7e84) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_RMDMNREQ\_V1 V1;
6. } DRS\_MSG\_RMDMNREQ;

**V1:**  The version 1 request.

##### DRS\_MSG\_RMDMNREQ\_V1

The DRS\_MSG\_RMDMNREQ\_V1 structure defines a request message sent to the [IDL\_DRSRemoveDsDomain](#Section_aa3cfa46c737425aae65ecaf9efe7e84) method.

1. typedef struct {
2. [string] LPWSTR DomainDN;
3. } DRS\_MSG\_RMDMNREQ\_V1;

**DomainDN:**  The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) to remove.

##### DRS\_MSG\_RMDMNREPLY

The DRS\_MSG\_RMDMNREPLY union defines the response messages received from the [IDL\_DRSRemoveDsDomain](#Section_aa3cfa46c737425aae65ecaf9efe7e84) method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_RMDMNREPLY\_V1 V1;
6. } DRS\_MSG\_RMDMNREPLY;

**V1:**  The version 1 response.

##### DRS\_MSG\_RMDMNREPLY\_V1

The DRS\_MSG\_RMDMNREPLY\_V1 structure defines a response message received from the [IDL\_DRSRemoveDsDomain](#Section_aa3cfa46c737425aae65ecaf9efe7e84) method.

1. typedef struct {
2. DWORD Reserved;
3. } DRS\_MSG\_RMDMNREPLY\_V1;

**Reserved:**  Unused. MUST be 0 and ignored.

#### Method-Specific Abstract Types and Procedures

##### HasNCReplicated

1. procedure HasNCReplicated(nc: DSName): boolean

Returns true if the [**DC's**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) of the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) specified by *nc* has replicated at least once with another DC that hosts that NC since the DC was booted; otherwise, returns false.

#### Server Behavior of the IDL\_DRSRemoveDsDomain Method

*Informative summary of behavior*: Removes the crossRef [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that defines a [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef). Fails if any [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) is currently hosting this [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) as its [**default NC**](#gt_adc2c434-9679-419f-8c8b-b8c234921ad3), as indicated by the state of that DC's nTDSDSA object. Fails if the server is not the Domain Naming [**FSMO role owner**](#gt_de81e9fd-25f5-4e90-aadb-1d35c5e8a06b) for the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62).

The removal of the crossRef object signals any DC currently hosting a partial [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) of the removed domain NC to remove that replica from its state.

This method undoes the effects of the IDL\_DRSAddEntry method when IDL\_DRSAddEntry is used to create a crossRef object.

The IDL\_DRSRemoveDsServer method removes the state within a forest, including the state on a DC's nTDSDSA object, associated with hosting a domain as a default NC on some DC. Therefore, IDL\_DRSRemoveDsServer can be used to establish a precondition for the success of IDL\_DRSRemoveDsDomain.

1. ULONG
2. IDL\_DRSRemoveDsDomain(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_RMDMNREQ \*pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DRS\_MSG\_RMDMNREPLY \*pmsgOut);
10. domainDN: unicodestring
11. otherNtdsdsa: DSName
12. cr: DSName
13. rt: ULONG
14. ValidateDRSInput(hDrs, 15)
15. pdwOutVersion^ := 1
16. pmsgOut^.V1.Reserved := 0
17. if dwInVersion ≠ 1 then
18. return ERROR\_INVALID\_PARAMETER
19. endif
20. domainDN := pmsgIn^.V1.DomainDN
21. if domainDN = null or domainDN = "" then
22. return ERROR\_INVALID\_PARAMETER
23. endif
24. /\* This function cannot be called on a DC for the domain
25. \* to be removed. \*/
26. if DefaultNC().dn = domainDN then
27. return ERROR\_DS\_ILLEGAL\_MOD\_OPERATION
28. endif
29. /\* Make sure no DCs still have NC replicas of this domain NC. \*/
30. otherNtdsdsa := select one o from ConfigNC() where
31. (nTDSDSA in o!objectClass) and
32. (domainDN in o!hasMasterNCs or
33. domainDN in o!msDS-hasMasterNCs)
34. if otherNtdsdsa ≠ null then
35. return ERROR\_DS\_NC\_STILL\_HAS\_DSAS
36. endif
37. /\* Find the crossRef object for the domain named by domainDN. \*/
38. cr := select one o from ConfigNC() where
39. (o!nCName = domainDN) and (crossRef in o!objectClass)
40. if cr = null then
41. return ERROR\_DS\_NO\_CROSSREF\_FOR\_NC
42. endif
43. /\* Make sure we are the Domain Naming FSMO role owner \*/
44. if GetFSMORoleOwner(FSMO\_DOMAIN\_NAMING) ≠ DSAObj()) then
45. /\* We are not the Domain Naming FSMO role owner \*/
46. return ERROR\_DS\_OBJ\_NOT\_FOUND
47. else
48. /\* We are the Domain Naming FSMO role owner. If the Config NC
49. \* has not replicated at least once since startup, our ownership
50. \* of the NC is not considered to be verified, so exit
51. \* with an error. \*/
52. if not HasNCReplicated(ConfigNC()) then
53. return ERROR\_DS\_ROLE\_NOT\_VERIFIED;
54. endif
55. endif
56. if (not AccessCheckObject(cr, RIGHT\_DELETE)) and
57. (not AccessCheckObject(cr.parent, RIGHT\_DS\_DELETE\_CHILD)) then
58. return ERROR\_ACCESS\_DENIED
59. endif
60. rt:= RemoveObj(cr,false)
61. if rt ≠ 0 then
62. return rt
63. endif
64. DelSubRef(cr!ncName)
65. return 0

### IDL\_DRSRemoveDsServer (Opnum 14)

The IDL\_DRSRemoveDsServer method removes the representation (also known as metadata) of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) from the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9).

1. ULONG IDL\_DRSRemoveDsServer(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_RMSVRREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_RMSVRREPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message. Must be set to 1 because that is the only version supported.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message. The value must be 1 because that is the only version supported.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if a failure occurs.

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_RMSVRREQ

The DRS\_MSG\_RMSVRREQ union defines the request messages sent to the [IDL\_DRSRemoveDsServer](#Section_d5c310ae347a49d4a78e6ffb2eecd581) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_RMSVRREQ\_V1 V1;
6. } DRS\_MSG\_RMSVRREQ;

**V1:**  The version 1 request.

##### DRS\_MSG\_RMSVRREQ\_V1

The DRS\_MSG\_RMSVRREQ\_V1 structure defines a request message sent to the [IDL\_DRSRemoveDsServer](#Section_d5c310ae347a49d4a78e6ffb2eecd581) method.

1. typedef struct {
2. [string] LPWSTR ServerDN;
3. [string] LPWSTR DomainDN;
4. BOOL fCommit;
5. } DRS\_MSG\_RMSVRREQ\_V1;

**ServerDN:**  The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the server [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) to remove.

**DomainDN:**  The DN of the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) that the DC to be removed belongs to. Can be set to null if the client does not want the server to compute the value of pmsgOut^.V1.fLastDCInDomain.

**fCommit:**  True if the DC's metadata should actually be removed from the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9). False if the metadata should not be removed. (This is used by a client that wants to determine the value of pmsgOut^.V1.fLastDcInDomain without altering the directory.)

##### DRS\_MSG\_RMSVRREPLY

The DRS\_MSG\_RMSVRREPLY union defines the response messages received from the [IDL\_DRSRemoveDsServer](#Section_d5c310ae347a49d4a78e6ffb2eecd581) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_RMSVRREPLY\_V1 V1;
6. } DRS\_MSG\_RMSVRREPLY;

**V1:**  The version 1 response.

##### DRS\_MSG\_RMSVRREPLY\_V1

The DRS\_MSG\_RMSVRREPLY\_V1 structure defines a response message received from the [IDL\_DRSRemoveDsServer](#Section_d5c310ae347a49d4a78e6ffb2eecd581) method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef struct {
2. BOOL fLastDcInDomain;
3. } DRS\_MSG\_RMSVRREPLY\_V1;

**fLastDcInDomain:**  True if the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) is the last DC in its [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), and pmsgIn^.V1.DomainDN was set to the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the domain to which the DC belongs. Otherwise, false.

#### Server Behavior of the IDL\_DRSRemoveDsServer Method

*Informative summary of behavior*: Removes the metadata defining a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), which consists of the tree of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) rooted at the DC's nTDSDSA object as well as the rIDSet objects and DRS [**SPNs**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) associated with the DC's computer object. This method is typically used when a DC is demoted. As part of the demotion process, the DC being demoted calls this method on another DC (either in the same [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), if such a DC exists, or in the parent domain, if there are no other DCs in the same domain but there is a parent domain) to remove the metadata of the DC being demoted from the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). Alternatively, if a DC is removed from the domain without being properly demoted (for example, if the DC suffers a fatal hardware failure), a client can make this call to remove the metadata of the now-nonexistent DC. When pmsgIn^.V1.DomainDN is specified, this method also computes whether the DC is the last [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) of its default [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef).

The behavior of this method has two variants. If pmsgIn^.V1.fCommit is false, the method is read-only with regard to abstract state; that is, it does not make any changes to the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) contents. In this mode, the main purpose of the method is to compute *pmsgOut^.V1.fLastDcInDomain* (and so there is little point to calling the method in this mode without setting pmsgIn^.V1.DomainDN). For example, prior to removing the DC's metadata, a client application might try to determine whether any DCs would be left in the domain, so that it can warn the user if the user is removing the last DC in the domain.

When pmsgIn^.V1.fCommit is true, the second variant of the behavior is performed. In this mode, the method actually removes the DC metadata. *The* pmsgOut^.V1.fLastDcInDomain value is also computed in this mode (provided that pmsgIn^.V1.DomainDN was passed in). This method undoes the effects of the IDL\_DRSAddEntry method when IDL\_DRSAddEntry is used to create an nTDSDSA object. The removal of the DC's metadata signals other DCs in the forest that this particular DC no longer exists.

1. ULONG
2. IDL\_DRSRemoveDsServer(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_RMSVRREQ \*pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DRS\_MSG\_RMSVRREPLY \*pmsgOut);
10. serverDn: unicodestring
11. domainDn: unicodestring
12. server: DSName
13. ntdsdsa: DSName
14. otherNtdsdsa: DSName
15. spnsToRemove: set of unicodestring
16. computerDn: unicodestring
17. computer: DSName
18. objectsToDelete: set of DSName
19. rt: ULONG
20. RODCKrbtgtAcct: DSName
21. accountList: set of DSName
22. ValidateDRSInput(hDrs, 14)
23. serverDn := pmsgIn^.V1.ServerDN
24. domainDn := pmsgIn^.V1.DomainDN
25. pdwOutVersion^ := 1
26. pmsgOut^.V1.fLastDcInDomain = false
27. /\* Basic parameter validation \*/
28. if dwInVersion ≠ 1 then
29. return ERROR\_INVALID\_PARAMETER
30. endif
31. if serverDn = null or serverDn = "" then
32. return ERROR\_INVALID\_PARAMETER
33. endif
34. /\* Note that DomainDN can be null, but it cannot be empty. \*/
35. if domainDn = "" then
36. return ERROR\_INVALID\_PARAMETER
37. endif
38. /\* Compute fLastDcInDomain if domainDn is non-null. \*/
39. if domainDn ≠ null then
40. otherNtdsdsa := select one o from subtree ConfigNC() where
41. (o!objectCategory = nTDSDSA)
42. and
43. (domainDn in o!hasMasterNCs or domainDn in o!msDS-hasMasterNCs)
44. and
45. (o ≠ ntdsdsa)
46. if otherNtdsdsa = null then
47. pmsgOut^.V1.fLastDcInDomain = true
48. else
49. pmsgOut^.V1.fLastDcInDomain = false
50. endif
51. endif
52. /\* If nothing to commit, processing is complete. \*/
53. if not pmsgIn^.V1.fCommit then
54. return 0
55. endif
56. ntdsdsa := DescendantObject([dn: serverDn], "CN=NTDS Settings,")
57. if ntdsdsa = null then
58. return ERROR\_DS\_CANT\_FIND\_DSA\_OBJ
59. endif
60. /\* Perform the actual DC metadata removal. \*/
61. /\* Locate the computer object for the DC's account. \*/
62. server := ntdsdsa!parent
63. computerDn := server!serverReference
64. computer := null
65. if computerDn ≠ null then
66. computer := GetDSNameFromDN(computerDn)
67. endif
68. /\* Remove the subtree of objects rooted at the DC's ntdsDsa object.\*/
69. if not AccessCheckObject(ntdsdsa, RIGHT\_DS\_DELETE\_TREE) then
70. return ERROR\_ACCESS\_DENIED
71. endif
72. rt := RemoveObj(ntdsdsa,true)
73. if rt ≠ 0 then
74. return rt
75. endif
76. /\* If the DC's computer account exists, remove rIDSet objects and
77. \* remove the DRS SPNs from the computer object. \*/
78. if computer ≠ null then
79. foreach r in computer!rIDSetReferences
80. if (not AccessCheckObject(r, RIGHT\_DELETE)) and
81. (not AccessCheckObject(r.parent, RIGHT\_DS\_DELETE\_CHILD)) then
82. return ERROR\_ACCESS\_DENIED
83. endif
84. RemoveObj(r, false)
85. endfor
86. foreach spn in computer!servicePrincipalName
87. if StartsWith(spn, "ldap/") or
88. StartsWith(spn, "GC/") or
89. StartsWith(spn, "E3514235-4B06-11D1-AB04-00C04FC2DCD2/") or
90. StartsWith(spn, "RPC/") then
91. spnsToRemove := spnsToRemove + {spn}
92. endif
93. endfor

/\* Cleanup for read-only domain controllers \*/

/\* Clear the KrbTgtLink from computer and delete its object \*/

/\* Get the msDS-KrbTgtLink attribute from the object \*/

RODCKrbtgtAcct := computer!msDS-KrbTgtLink

/\* Delet the attribute from the object \*/

Computer!msDS-KrbTgtLink := null

/\* Remove the KrbTgtLink \*/

RemoveObj(RODCKrbTgtLink, false)

/\* Delete RODC policies \*/

computer!msDS-NeverRevealGroup := null

computer!msDS-RevealOnDemandGroup := null

computer!msDS-RevealedUsers := null

/\* Delete msDS-AuthenticatedToAccountList links \*/

accountList := { computer!msDS-AuthenticatedToAccountList }

foreach entry in accountList

entry!msDS-AuthenticatedAtDC := entry!msDS-AuthenticatedAtDC – computer

endfor

1. if not AccessCheckAttr(computer, servicePrincipalName,
2. RIGHT\_DS\_WRITE\_PROPERTY) then
3. return ERROR\_ACCESS\_DENIED
4. endif
5. computer!servicePrincipalName :=
6. computer!servicePrincipalName - spnsToRemove
7. endif
8. return 0

### IDL\_DRSReplicaAdd (Opnum 5)

The IDL\_DRSReplicaAdd method adds a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) source reference for the specified [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942).

1. ULONG IDL\_DRSReplicaAdd(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwVersion,
4. [in, ref, switch\_is(dwVersion)]
5. DRS\_MSG\_REPADD\* pmsgAdd
6. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwVersion:** The version of the request message.

**pmsgAdd:** A pointer to the request message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_REPADD

The DRS\_MSG\_REPADD union defines request messages that are sent to the [IDL\_DRSReplicaAdd](#Section_7219df914eea494f88e3780d40d2d559) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_REPADD\_V1 V1;
6. [case(2)]
7. DRS\_MSG\_REPADD\_V2 V2;
8. [case(3)]
9. DRS\_MSG\_REPADD\_V3 V3;
10. } DRS\_MSG\_REPADD;

**V1:**  The version 1 request.

**V2:**  The version 2 request (a superset of V1).

**V3**: The version 3 request (a superset of V2).

##### DRS\_MSG\_REPADD\_V1

The DRS\_MSG\_REPADD\_V1 structure defines a request message sent to the [IDL\_DRSReplicaAdd](#Section_7219df914eea494f88e3780d40d2d559) method.

1. typedef struct {
2. [ref] DSNAME\* pNC;
3. [ref] [string] char\* pszDsaSrc;
4. REPLTIMES rtSchedule;
5. ULONG ulOptions;
6. } DRS\_MSG\_REPADD\_V1;

**pNC:**  The [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the NC to replicate.

**pszDsaSrc:**  The transport-specific [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) from which to replicate [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493).

**rtSchedule:**  The schedule used to perform periodic [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb).

**ulOptions:**  Zero or more [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) flags.

##### DRS\_MSG\_REPADD\_V2

The DRS\_MSG\_REPADD\_V2 structure defines a request message sent to the [IDL\_DRSReplicaAdd](#Section_7219df914eea494f88e3780d40d2d559) method. This request version is a superset of V1.

1. typedef struct {
2. [ref] DSNAME\* pNC;
3. [unique] DSNAME\* pSourceDsaDN;
4. [unique] DSNAME\* pTransportDN;
5. [ref] [string] char\* pszSourceDsaAddress;
6. REPLTIMES rtSchedule;
7. ULONG ulOptions;
8. } DRS\_MSG\_REPADD\_V2;

**pNC:**  The [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the NC to replicate.

**pSourceDsaDN:**  The nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) from which to replicate changes.

**pTransportDN:**  The interSiteTransport object that identifies the network transport over which [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) is to be performed.

**pszSourceDsaAddress:**  The transport-specific [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) of the DC from which to replicate [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493).

**rtSchedule:**  The schedule used to perform periodic replication.

**ulOptions:**  Zero or more [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) flags.

##### DRS\_MSG\_REPADD\_V3

The DRS\_MSG\_REPADD\_V3 structure defines a request message sent to the [IDL\_DRSReplicaAdd](#Section_7219df914eea494f88e3780d40d2d559) method. This request version is a superset of V2.

1. typedef struct {
2. [ref] DSNAME\* pNC;
3. [unique] DSNAME\* pSourceDsaDN;
4. [unique] DSNAME\* pTransportDN;
5. [ref] [string] char\* pszSourceDsaAddress;
6. REPLTIMES rtSchedule;
7. ULONG ulOptions;
8. GUID correlationID;
9. [unique] VAR\_SIZE\_BUFFER\_WITH\_VERSION\* pReservedBuffer;
10. } DRS\_MSG\_REPADD\_V3;

**pNC:**  The [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the NC to replicate.

**pSourceDsaDN:**  The nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) from which to replicate changes.

**pTransportDN:**  The interSiteTransport object that identifies the network transport over which [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) is to be performed.

**pszSourceDsaAddress:**  The transport-specific [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) of the DC from which to replicate [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493).

**rtSchedule:**  The schedule used to perform periodic replication.

**ulOptions:**  Zero or more [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) flags.

**correlationID**: An identifier for the operation that the DC can use for implementation-defined troubleshooting. There are no normative constraints on this value, nor does the value figure in any normative processing rules.

**pReservedBuffer**: A pointer to a VAR\_SIZE\_BUFFER\_WITH\_VERSION structure (section [5.219](#Section_589574c1eaa1456fac53de597b2cff6b)). MUST be a null pointer.

#### Server Behavior of the IDL\_DRSReplicaAdd Method

*Informative summary of behavior*: The server adds a value to the repsFrom of the specified [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). If ulOptions contains DRS\_ASYNC\_OP, the server processes the request asynchronously. The client can be an administrative client or another [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). The client includes DRS\_WRIT\_REP in ulOptions if the specified NC replica is writable at the server. The client includes DRS\_NONGC\_RO\_REP and DRS\_SPECIAL\_SECRET\_PROCESSING in ulOptions if the specified NC replica is a read-only full [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) on a [**read-only DC**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870). The server adds a value to [repsFrom](#Section_3ef27d3cb9c944048e53ebf3a64a9a10), and the value has replicaFlags derived from ulOptions (see below), serverAddress equal to pszSourceDsaAddress (pszDsaSrc if V1), and schedule equal to rtSchedule. If ulOptions contains DRS\_ASYNC\_REP but not DRS\_MAIL\_REP or DRS\_NEVER\_NOTIFY, the server sends a request to the DC specified by pszSourceDsaAddress to add a value to the repsTo of the specified NC replica by calling IDL\_DRSUpdateRefs. Finally, the server begins a [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16) by sending an IDL\_DRSGetNCChanges request.

1. ULONG
2. IDL\_DRSReplicaAdd(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwVersion,
5. [in, ref, switch\_is(dwVersion)] DRS\_MSG\_REPADD \*pmsgAdd);
6. options: DRS\_OPTIONS
7. nc: DSName
8. partitionsObj: DSName
9. cr: DSName
10. rf: RepsFrom
11. msgIn: DRS\_MSG\_REPADD\_V2
12. updRefs: DRS\_MSG\_UPDREFS /\* See IDL\_DRSUpdateRefs structures. \*/
13. hDrsSrc: DRS\_HANDLE
14. msgRequest: DRS\_MSG\_GETCHGREQ
15. msgOut: DRS\_MSG\_GETCHGREPLY
16. outVersion: DWORD
17. cMaxObjects: ULONG
18. cMaxBytes: ULONG
19. versionRequestMsg: DWORD
20. err: ULONG
21. ValidateDRSInput(hDrs, 5)
22. /\* Validate the version \*/
23. if dwVersion ≠ 1 and dwVersion ≠ 2 then
24. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
25. endif
26. if dwVersion = 1 then
27. msgIn := pmsgAdd^.V1
28. msgIn.pszSourceDsaAddress = pmsgAdd^.V1.pszDsaSrc
29. else
30. msgIn := pmsgAdd^.V2
31. endif
32. if msgIn.pNC = null
33. or msgIn.pszSourceDsaAddress = null
34. or msgIn.pszSourceDsaAddress = "" then
35. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
36. endif
37. options := msgIn.ulOptions
38. nc := msgIn.pNC^
39. partitionsObj :=
40. select one o from children ConfigNC() where o!name = "Partitions"
41. cr := select o from children partitionsObj where o!nCName = nc
42. if cr = null then
43. return ERROR\_DS\_DRA\_BAD\_NC
44. endif
45. if options - {DRS\_ASYNC\_OP, DRS\_CRITICAL\_ONLY, DRS\_ASYNC\_REP,
46. DRS\_WRIT\_REP, DRS\_INIT\_SYNC, DRS\_PER\_SYNC, DRS\_MAIL\_REP,
47. DRS\_NONGC\_RO\_REP, DRS\_SPECIAL\_SECRET\_PROCESSING, DRS\_DISABLE\_AUTO\_SYNC,
48. DRS\_DISABLE\_PERIODIC\_SYNC, DRS\_USE\_COMPRESSION, DRS\_NEVER\_NOTIFY,
49. DRS\_TWOWAY\_SYNC} ≠ {} then
50. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
51. endif
52. if AmIRODC() and DRS\_WRIT\_REP in options then
53. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
54. endif
55. if AmIRODC() and DRS\_MAIL\_REP in options then
56. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
57. endif
58. if DRS\_MAIL\_REP in options and not DRS\_ASYNC\_REP in options then
59. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
60. endif
61. if ObjExists(nc) then
62. if not AccessCheckCAR(nc, DS-Replication-Manage-Topology) then
63. return ERROR\_DS\_DRA\_ACCESS\_DENIED
64. endif
65. else
66. if not AccessCheckCAR(DefaultNC(), DS-Replication-Manage-Topology)
67. then
68. return ERROR\_DS\_DRA\_ACCESS\_DENIED
69. endif
70. endif
71. if DRS\_ASYNC\_OP in options then
72. Asynchronous Processing: Initiate a logical thread of control
73. to process the remainder of this request asynchronously
74. return 0
75. endif
76. if ObjExists(nc) then
77. if (IT\_WRITE in nc!instanceType) ≠ (DRS\_WRIT\_REP in options) then
78. return ERROR\_DS\_DRA\_BAD\_INSTANCE\_TYPE
79. endif
80. /\* Disallow addition if server already replicates from this
81. \* source \*/
82. if (select one v from nc!repsFrom
83. where v.serverAddress = msgIn.pszSourceDsaAddress) ≠ null
84. then
85. return ERROR\_DS\_DRA\_DN\_EXISTS
86. endif
87. endif
88. if DRS\_ASYNC\_REP in options then
89. if msgIn.pSourceDsaDN = null
90. or not ObjExists(msgIn.pSourceDsaDN^)
91. then
92. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
93. endif
94. endif
95. if DRS\_MAIL\_REP in options then
96. if msgIn.pTransportDN = null
97. or not ObjExists(msgIn.pTransportDN^)
98. then
99. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
100. endif
101. endif
102. /\* Construct RepsFrom value. \*/
103. if msgIn.pSourceDsaDN ≠ null then
104. rf.uuidDsa := msgIn.pSourceDsaDN^!objectGUID
105. endif
106. if msgIn.pTransportDN ≠ null then
107. rf.uuidTransportObj := msgIn.pTransportDN^!objectGUID
108. endif
109. rf.replicaFlags := msgIn.ulOptions ∩ {DRS\_DISABLE\_AUTO\_SYNC,
110. DRS\_DISABLE\_PERIODIC\_SYNC, DRS\_INIT\_SYNC, DRS\_MAIL\_REP,
111. DRS\_NEVER\_NOTIFY, DRS\_PER\_SYNC, DRS\_TWOWAY\_SYNC,
112. DRS\_USE\_COMPRESSION, DRS\_WRIT\_REP, DRS\_NONGC\_RO\_REP,
113. DRS\_SPECIAL\_SECRET\_PROCESSING }
114. rf.schedule := msgIn.rtSchedule^
115. rf.serverAddress := msgIn.pszSourceDsaAddress^
116. rf.timeLastAttempt := current time
117. nc!repsFrom := nc!repsFrom + {rf}
118. if msgIn.ulOptions ∩ {DRS\_ASYNC\_REP, DRS\_NEVER\_NOTIFY, DRS\_MAIL\_REP}
119. = {DRS\_ASYNC\_REP} then
120. /\* Enable replication notifications by requesting the server DC
121. \* to add a repsTo for this DC. \*/
122. updRefs.pNC^ := ADR(nc)
123. updRefs.pszDsaDest := NetworkAddress of this DC
124. updRefs.uuidDsaObjDest := dc.serverGuid
125. updRefs.ulOptions := {DRS\_ASYNC\_OP, DRS\_ADD\_REF, DRS\_DEL\_REF}
126. if DRS\_WRIT\_REP in msgIn.ulOptions then
127. updRefs.ulOptions := updRefs.ulOptions + {DRS\_WRIT\_REP}
128. endif
129. hDrsSrc := BindToDSA(msgIn.pSourceDsaDN)
130. if hDrsSrc ≠ null then
131. ret := IDL\_DRSUpdateRefs(hDrsSrc, 1, ADR(updRefs))
132. UnbindFromDSA(hDrsSrc)
133. endif
134. endif
135. /\* Perform a replication cycle as a client of IDL\_DRSGetNCChanges. \*/
136. versionRequestMsg := The version number of the input message negotiated between the
137. client and server (section 4.1.10.4.1).
138. cMaxObjects := Implementation-specific value.
139. cMaxBytes := Implementation-specific value.

/\* Form the first request \*/

ReplicateNCRequestMsg(

hDrsSrc,

versionRequestMsg,

nc,

rf,

options,

cMaxObjects,

cMaxBytes,

ADDR(msgRequest))

err := IDL\_DRSGetNCChanges(

hDrsSrc,

versionRequestMsg,

msgRequest,

ADDR(outVersion),

ADDR(msgOut))

if err = 0

and not DRS\_MAIL\_REP in msgIn.ulOptions

then

Wait for the response, process it (section 4.1.10.6), send the next request, etc.

until the replication cycle is complete.

If there are any failures from this replication attempt, err is assigned an

appropriate error value.

endif

1. return err

### IDL\_DRSReplicaDel (Opnum 6)

The IDL\_DRSReplicaDel method deletes a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) source reference for the specified [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942).

1. ULONG IDL\_DRSReplicaDel(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwVersion,
4. [in, ref, switch\_is(dwVersion)]
5. DRS\_MSG\_REPDEL\* pmsgDel
6. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d).

**dwVersion:** The version of the request message.

**pmsgDel:** A pointer to the request message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_REPDEL

The DRS\_MSG\_REPDEL union defines the request messages sent to the [IDL\_DRSReplicaDel](#Section_1420a9bf9267464da6d57676472d7f1d) method. Only one version, identified by *dwVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_REPDEL\_V1 V1;
6. } DRS\_MSG\_REPDEL;

**V1:**  The version 1 request.

##### DRS\_MSG\_REPDEL\_V1

The DRS\_MSG\_REPDEL\_V1 structure defines a request message sent to the [IDL\_DRSReplicaDel](#Section_1420a9bf9267464da6d57676472d7f1d) method.

1. typedef struct {
2. [ref] DSNAME\* pNC;
3. [string] char\* pszDsaSrc;
4. ULONG ulOptions;
5. } DRS\_MSG\_REPDEL\_V1;

**pNC:**  A pointer to [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the root of an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) on the server.

**pszDsaSrc:**  The transport-specific [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**ulOptions:**  The [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) flags.

#### Server Behavior of the IDL\_DRSReplicaDel Method

*Informative summary of behavior*: When DRS\_NO\_SOURCE is not specified, the server removes a value from the repsFrom of the specified [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). If ulOptions contains DRS\_ASYNC\_OP, the server processes the request asynchronously. The client has to include DRS\_WRIT\_REP in ulOptions if the specified NC replica is a writable [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac). The server removes the value from repsFrom whose serverAddress matches pszDsaSrc. If ulOptions does not contain DRS\_LOCAL\_ONLY, the server sends a request to the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) specified by pszDsaSrc to remove this DC from the values in repsTo of the specified NC replica by calling IDL\_DRSUpdateRefs.

When DRS\_NO\_SOURCE is specified, the server [**expunges**](#gt_c947d085-898e-44c0-a849-47c3b817b7b7) the NC replica and all its children. This operation returns an error and the expunge does not occur if the repsFrom or repsTo [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) are present on the NC replica. However, if ulOptions contains DRS\_REF\_OK, it is permitted for repsTo to be present. If ulOptions contains DRS\_ASYNC\_OP, the server processes the request asynchronously. The client has to include DRS\_WRIT\_REP in ulOptions if the specified NC replica is writable. If ulOptions contains DRS\_ASYNC\_REP, the server expunges the [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) asynchronously.

1. ULONG
2. IDL\_DRSReplicaDel(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwVersion,
5. [in, ref, switch\_is(dwVersion)] DRS\_MSG\_REPDEL \*pmsgDel);
6. options: DRS\_OPTIONS
7. nc: DSName
8. cr: DSName
9. srcDSA: DSName
10. hDrsSrc: DRS\_HANDLE
11. rf: RepsFrom
12. msgIn: DRS\_MSG\_REPDEL\_V1
13. updRefs: DRS\_MSG\_UPDREFS /\* See IDL\_DRSUpdateRefs structures. \*/
14. rt: ULONG
15. ValidateDRSInput(hDrs, 6)
16. msgIn := pmsgDel^.V1
17. /\* Validate the NC \*/
18. if msgIn.pNC = null then
19. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
20. endif
21. nc := msgIn.pNC^
22. if not ObjExists(nc) then
23. return ERROR\_DS\_DRA\_BAD\_NC
24. endif
25. if not AccessCheckCAR(nc, DS-Replication-Manage-Topology) then
26. return ERROR\_DS\_DRA\_ACCESS\_DENIED
27. endif
28. options := msgIn.ulOptions
29. /\* Any request that includes invalid options is rejected. \*/
30. if options - {DRS\_ASYNC\_OP, DRS\_WRIT\_REP, DRS\_MAIL\_REP,DRS\_ASYNC\_REP,
31. DRS\_IGNORE\_ERROR, DRS\_LOCAL\_ONLY, DRS\_NO\_SOURCE, DRS\_REF\_OK} ≠ {} then
32. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
33. endif
34. if DRS\_NO\_SOURCE in options then
35. /\* Expunging local copy of an NC. \*/
36. /\* Do not permit removal of nonroot or uninstantiated NCs. \*/
37. if (IT\_NC\_HEAD not in nc!instanceType or
38. IT\_UNINSTANT in nc!instanceType) then
39. return ERROR\_DS\_DRA\_BAD\_NC
40. endif
41. /\* NC must not replicate from any other DC. \*/
42. if (select one v from nc!repsFrom where (true)) ≠ null then
43. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
44. endif
45. /\* NC must not replicate to any other DC. \*/
46. if (select one v from nc!repsTo where (true)) ≠ null
47. and (not DRS\_REF\_OK in options) then
48. return ERROR\_DS\_DRA\_OBJ\_IS\_REP\_SOURCE
49. endif
50. /\* Do not permit removal of important NCs. \*/
51. if IT\_WRITE in nc!instanceType
52. and (nc = DefaultNC()
53. or nc = ConfigNC()
54. or nc = SchemaNC()) then
55. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
56. endif
57. if DRS\_ASYNC\_REP in options then
58. Asynchronous Processing: Initiate a logical thread of control
59. to process the remainder of this request asynchronously
60. return 0
61. endif
62. /\* Expunge the subtree rooted at dn and pertaining to the same NC.
63. \* If the subtree includes a sub-ref object for a locally instantiated NC,
64. \* remove the IT\_NC\_ABOVE flag from the sub-ref object instanceType
65. \* attribute.
66. \*
67. \*/
68. foreach o in (select all v from subtree nc where GetObjectNC(v) = nc)
69. if(IT\_NC\_HEAD in o!instanceType and
70. IT\_UNINSTANT not in o!instanceType) then
71. o!instanceType = o!instanceType – {IT\_NC\_ABOVE}
72. else
73. Expunge(o)
74. endif
75. endfor
76. /\* If the root of the NC being expunged is a sub-ref object itself, then it
77. \* might need to be preserved.
78. \*/
79. /\* Check whether there is stil a crossref object for the given nc. \*/
80. cr := select one v from subtree ConfigNC()
81. where v!ncName = nc and crossRef in v!objectClass
82. if(cr == NULL)
83. if(IT\_NC\_ABOVE in nc!instanceType) then
84. nc!instanceType = {IT\_NC\_ABOVE, IT\_UNINSTANT,IT\_NC\_HEAD}
85. endif
86. rt := RemoveObj(nc,false)
87. if rt ≠ 0 then
88. return rt
89. endif
90. else
91. if(IT\_NC\_ABOVE in nc!instanceType) then
92. nc!instanceType = {IT\_NC\_ABOVE, IT\_UNINSTANT,IT\_NC\_HEAD}
93. else
94. Expunge(nc)
95. endif
96. endif
97. return 0
98. else /\* not DRS\_NO\_SOURCE in options \*/
99. /\* Removing a single source from repsFrom, but leaving NC replica
100. \* on DC. \*/
101. if msgIn.pszDsaSrc = null or
102. msgIn.pszDsaSrc^ = "" or
103. (IsAdlds() and
104. GetDSNameFromNetworkAddress(msgIn.pszDsaSrc^) = null) then
105. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
106. endif
107. if DRS\_ASYNC\_OP in options then
108. Asynchronous Processing: Initiate a logical thread of control
109. to process the remainder of this request asynchronously
110. return 0
111. endif
112. rf := select one v from nc!repsFrom
113. where (v.serverAddress = msgIn.pszDsaSrc)
114. if rf = null then
115. return ERROR\_DS\_DRA\_NO\_REPLICA
116. endif
117. nc!repsFrom := nc!repsFrom - {rf}
118. if (not DRS\_LOCAL\_ONLY in options)
119. and (not DRS\_MAIL\_REP in rf.options) then
120. /\* Disable replication notifications by requesting the server DC
121. \* specified by msgIn.pszDsaSrc to remove this DC
122. \* from its repsTo. \*/
123. updRefs.pNC^ := ADR(nc)
124. updRefs.pszDsaDest := NetworkAddress of this DC
125. updRefs.uuidDsaDest := dc.serverGuid
126. updRefs.ulOptions := {DRS\_ASYNC\_OP, DRS\_DEL\_REF}
127. if DRS\_WRIT\_REP in msgIn.ulOptions then
128. updRefs.ulOptions := updRefs.ulOptions + {DRS\_WRIT\_REP}
129. endif
130. srcDSA := GetDSNameFromNetworkAddr(msgnIn.pszDsaSrc)
131. hDrsSrc := BindToDSA(srcDSA)
132. if hDrsSrc ≠ null then
133. ret := IDL\_DRSUpdateRefs(hDrsSrc, 1, ADR(updRefs))
134. UnbindFromDSA(hDrsSrc)
135. endif
136. endif
137. return 0
138. endif

### IDL\_DRSReplicaDemotion (Opnum 26)

The IDL\_DRSReplicaDemotion method replicates off all changes to the specified [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) and moves any [**FSMOs**](#gt_3fcc9e5e-60b6-40f8-acb6-ad3189cf90ec) held to another server.

1. ULONG IDL\_DRSReplicaDemotion(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_REPLICA\_DEMOTIONREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_REPLICA\_DEMOTIONREPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_REPLICA\_DEMOTIONREQ

The DRS\_MSG\_REPLICA\_DEMOTIONREQ union defines the request messages sent to the [IDL\_DRSReplicaDemotion](#Section_8a2f0388bdfb4519a8c3384f27c11639) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_REPLICA\_DEMOTIONREQ\_V1 V1;
6. } DRS\_MSG\_REPLICA\_DEMOTIONREQ;

**V1:**  The version 1 request. Only one version is defined.

##### DRS\_MSG\_REPLICA\_DEMOTIONREQ\_V1

The DRS\_MSG\_REPLICA\_DEMOTIONREQ\_V1 structure defines a request message sent to the [IDL\_DRSReplicaDemotion](#Section_8a2f0388bdfb4519a8c3384f27c11639) method.

1. typedef struct {
2. DWORD dwFlags;
3. UUID uuidHelperDest;
4. [ref] DSNAME\* pNC;
5. } DRS\_MSG\_REPLICA\_DEMOTIONREQ\_V1;

**dwFlags:**  Zero or more of the following bit flags, which are presented in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | X | X | X | T | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero.

**T (DS\_REPLICA\_DEMOTE\_TRY\_ALL\_SRCS, 0x00000001)**: MUST be set.

**uuidHelperDest:**  Unused. Must be [**NULL GUID**](#gt_ba500a5b-8c29-467c-a335-0980c8b11304) and ignored.

**pNC:**  The [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) of the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) to replicate off.

##### DRS\_MSG\_REPLICA\_DEMOTIONREPLY

The DRS\_MSG\_REPLICA\_DEMOTIONREPLY union defines the response messages received from the [IDL\_DRSReplicaDemotion](#Section_8a2f0388bdfb4519a8c3384f27c11639) method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_REPLICA\_DEMOTIONREPLY\_V1 V1;
6. } DRS\_MSG\_REPLICA\_DEMOTIONREPLY;

V1:  The version 1 reply.

##### DRS\_MSG\_REPLICA\_DEMOTIONREPLY\_V1

The DRS\_MSG\_REPLICA\_DEMOTIONREPLY\_V1 structure defines a response message received from the [IDL\_DRSReplicaDemotion](#Section_8a2f0388bdfb4519a8c3384f27c11639) method.

1. typedef struct {
2. DWORD dwOpError;
3. } DRS\_MSG\_REPLICA\_DEMOTIONREPLY\_V1;

**dwOpError:**  The Win32 error code, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.2.

#### Method-Specific Abstract Types and Procedures

##### ReplicationPartners()

1. procedure ReplicationPartners(nc: DSNAME): sequence of DSNAME

The [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) D executing this procedure hosts a portion of [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) F. This procedure computes the set of all DCs in F that host the specified [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942), excluding D. It returns this set as a sequence in an arbitrary order.

##### AbandonAllFSMORoles()

1. procedure AbandonAllFSMORoles(nc: DSNAME): DWORD

The AbandonAllFSMORoles procedure abandons any [**FSMO roles**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f) represented in the supplied [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) that are held by this [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). The new holder of the FSMO roles is arbitrary. AbandonAllFSMORoles returns a Win32 error value.

1. targetDSAs: sequence of DSNAME
2. fsmoContainer: DSNAME
3. ret: DWORD
4. bGivenAway: boolean
5. i: integer
6. hDRS: DRS\_HANDLE
7. msgReq: DRS\_MSG\_GETCHGREQ\_V10
8. msgUpd: DRS\_MSG\_GETCHGREPLY\_NATIVE
9. if nc = ConfigNC() then
10. /\* check domain naming FSMO role \*/
11. fsmoContainer := DescendantObject(ConfigNC(), "CN=Partitions,")
12. else if nc = SchemaNC() then
13. /\* check schema master FSMO role \*/
14. fsmoContainer := SchemaNC()
15. else
16. /\* application NCs don't hold FSMOs \*/
17. return ERROR\_SUCCESS
18. endif
19. /\* check if we hold the fsmo \*/
20. if fsmoContainer!fSMORoleOwner ≠ DSAObj() then
21. /\* we do not own the role! All's well \*/
22. return ERROR\_SUCCESS
23. endif
24. /\* yes, we own the role! Let's give it away \*/
25. bGivenAway := false
26. targetDSAs := ReplicationPartners(nc)
27. i := 0
28. while not bGivenAway
29. if i ≥ targetDSAs.length then
30. /\* no more replication partners that would take our FSMO! \*/
31. return ERROR\_DS\_UNABLE\_TO\_SURRENDER\_ROLES
32. endif
33. hDRS := BindToDSA(targetDSAs[i])
34. if hDRS ≠ null then
35. /\* the targetDSA appears to be up. Let's try to transfer the
36. \* role \*/
37. /\* Perform an IDL\_DRSGetNCChanges(EXOP\_FSMO\_ABANDON\_ROLE) call \*/
38. msgReq.uuidDsaObjDest := dc.serverGuid
39. msgReq.pNC := ADDR(fsmoContainer)
40. msgReq.ulFlags := DRS\_WRIT\_REP
41. msgReq.ulExtendedOp := EXOP\_FSMO\_ABANDON\_ROLE
42. ret :=
43. IDL\_DRSGetNCChanges(hDRS, 8, ADDR(msgReq), 6, ADDR(msgUpd))
44. if ret = ERROR\_SUCCESS then
45. /\* successfully given away \*/
46. bGivenAway := true
47. endif
48. UnbindFromDSA(hDRS)
49. endif
50. i := i + 1
51. endwhile
52. /\* if execution got here, the role was given away \*/
53. return ERROR\_SUCCESS

##### ReplicateOffChanges()

1. procedure ReplicateOffChanges(nc: DSNAME): DWORD

The ReplicateOffChanges procedure replicates all local changes in the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) to a randomly selected [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) partner.

1. targetDSAs: sequence of DSNAME
2. ret: DWORD
3. bReplicated: boolean
4. i: integer
5. msgSyncReq: DRS\_MSG\_REPSYNC\_V1
6. msgAddReq: DRS\_MSG\_REPADD\_V2
7. hDRS: DRS\_HANDLE
8. bReplicated := false
9. targetDSAs := ReplicationPartners(nc)
10. i := 0
11. while not bReplicated
12. if i ≥ targetDSAs.length then
13. /\* no more replication partners that host the NC! \*/
14. return ERROR\_DS\_CANT\_FIND\_DSA\_OBJ
15. endif
16. hDRS := BindToDSA(targetDSAs[i])
17. if hDRS ≠ null then
18. /\* the targetDSA appears to be up. Let's try to replicate to
19. \* it \*/
20. /\* Invoke IDL\_DRSReplicaSync to get changes from us \*/
21. msgSyncReq.pszDsaSrc := NetworkAddress of targetDSA
22. msgSyncReq.uuidDsaSrc := dc.serverGuid
23. msgSyncReq.pNC := ADDR(nc)
24. msgSyncReq.ulOptions := DRS\_WRIT\_REP
25. ret := IDL\_DRSReplicaSync(hDRS, 1,ADR(msgSyncReq))
26. if ret = ERROR\_DS\_DRA\_NO\_REPLICA then
27. /\* the targetDSA does not currently have replication agreement
28. (repsFrom) with this DC. Tell it to add one \*/
29. msgAddReq.pNC := ADDR(nc)
30. msgAddReq.pszSourceDsaAddress := NetworkAddress of this DC
31. msgAddReq.ulOptions := DRS\_WRIT\_REP
32. msgAddReq.pSourceDsaDN := null
33. msgAddReq.pTransportDN := null
34. ret := IDL\_DRSReplicaAdd(hDRS, 2,ADR(msgAddReq))
35. endif
36. UnbindFromDSA(hDRS)
37. if ret = ERROR\_SUCCESS then
38. /\* success! \*/
39. bReplicated := true
40. endif
41. endif
42. i := i + 1
43. endwhile
44. /\* if execution got here, then the changes were successfully replicated off \*/
45. return ERROR\_SUCCESS

#### Server Behavior of the IDL\_DRSReplicaDemotion Method

*Informative summary of behavior*: For a given [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942), the [IDL\_DRSReplicaDemotion](#Section_8a2f0388bdfb4519a8c3384f27c11639) method replicates out any changes that had not previously been replicated out. It also abandons any NC-specific [**FSMO roles**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f) that are owned by this [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). This function accomplishes nothing when the DC being demoted is the last DC in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62).

1. ULONG
2. IDL\_DRSReplicaDemotion(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_REPLICA\_DEMOTIONREQ\* pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DRS\_MSG\_REPLICA\_DEMOTIONREPLY\* pmsgOut
10. )
11. msgIn: DRS\_MSG\_REPLICA\_DEMOTIONREQ\_V1
12. ret: DWORD
13. nc: DSNAME
14. ValidateDRSInput(hDrs, 26)
15. pdwOutVersion^ := 1
16. pmsgOut^.V1.dwOpError := ERROR\_DS\_CODE\_INCONSISTENCY
17. if dwInVersion ≠ 1 then
18. return ERROR\_INVALID\_PARAMETER
19. endif
20. msgIn := pmsgIn^.V1
21. if msgIn.pNC = null or
22. msgIn.dwFlags ≠ DS\_REPLICA\_DEMOTE\_TRY\_ALL\_SRCS then
23. return ERROR\_INVALID\_PARAMETER
24. endif
25. if not IsMemberOfBuiltinAdminGroup() then
26. /\* only BA is allowed to demote an AD LDS service \*/
27. return ERROR\_DS\_DRA\_ACCESS\_DENIED
28. endif
29. nc := msgIn.pNC^
30. ret := AbandonAllFSMORoles(nc)
31. if ret = ERROR\_SUCCESS then
32. ret := ReplicateOffChanges(nc)
33. endif
34. if ret = ERROR\_SUCCESS then
35. /\* mark instanceType as going and not coming \*/
36. nc!instanceType := nc!instanceType + {IT\_NC\_GOING} - {IT\_NC\_COMING}
37. /\* remove any repsFrom \*/
38. nc!repsFrom := null
39. endif
40. pmsgOut^.V1.dwOpError := ret
41. pdwMsgOut^ := 1
42. return ERROR\_SUCCESS

### IDL\_DRSReplicaModify (Opnum 7)

The IDL\_DRSReplicaModify method [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the value for repsFrom for the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

1. ULONG IDL\_DRSReplicaModify(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwVersion,
4. [in, ref, switch\_is(dwVersion)]
5. DRS\_MSG\_REPMOD\* pmsgMod
6. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d).

**dwVersion:** The version of the request message.

**pmsgMod:** A pointer to the request message.

**Return Values:** 0 if successful, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if a failure occurs.

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_REPMOD

The DRS\_MSG\_REPMOD union defines the request messages for the [IDL\_DRSReplicaModify](#Section_cd241bf256be453786b1cdbc997b0860) method. Only one version, identified by *dwVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_REPMOD\_V1 V1;
6. } DRS\_MSG\_REPMOD;

**V1:**  The version 1 request.

##### DRS\_MSG\_REPMOD\_V1

The DRS\_MSG\_REPMOD\_V1 structure defines a request message for the [IDL\_DRSReplicaModify](#Section_cd241bf256be453786b1cdbc997b0860) method.

1. typedef struct {
2. [ref] DSNAME\* pNC;
3. UUID uuidSourceDRA;
4. [unique, string] char\* pszSourceDRA;
5. REPLTIMES rtSchedule;
6. ULONG ulReplicaFlags;
7. ULONG ulModifyFields;
8. ULONG ulOptions;
9. } DRS\_MSG\_REPMOD\_V1;

**pNC:**  A pointer to the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the root of an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) on the server.

**uuidSourceDRA:**  The [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1).

**pszSourceDRA:**  The transport-specific [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**rtSchedule:**  The periodic [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) schedule.

**ulReplicaFlags:**  The [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) flags for the repsFrom value.

**ulModifyFields:**  The fields to [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493)(presented in little-endian byte order).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | X | U S | U A | U F | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**UF (DRS\_UPDATE\_FLAGS, 0x00000001)**: Updates the flags associated with the server.

**UA (DRS\_UPDATE\_ADDRESS, 0x00000002)**: Updates the transport-specific address associated with the server.

**US (DRS\_UPDATE\_SCHEDULE, 0x00000004)**: Updates the replication schedule associated with the server.

**ulOptions:**  The DRS\_OPTIONS flags for execution of this method.

#### Server Behavior of the IDL\_DRSReplicaModify Method

*Informative summary of behavior*: The server replaces fields in the repsFrom of the specified [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). If ulOptions contains DRS\_ASYNC\_OP, the server processes the request asynchronously. The client has to include DRS\_WRIT\_REP in ulOptions if the specified NC replica is a full [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac). The server optionally replaces (as specified by ulModifyFields) serverAddress, schedule, and replicaFlags in repsFrom with the corresponding value from pszSourceDRA, rtSchedule, and ulReplicaFlags.

1. ULONG
2. IDL\_DRSReplicaModify(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwVersion,
5. [in, ref, switch\_is(dwVersion)]
6. DRS\_MSG\_REPMOD \*pmsgMod);
7. options: DRS\_OPTIONS
8. nc: DSName
9. rf: RepsFrom
10. msgIn: DRS\_MSG\_REPMOD\_V1
11. ValidateDRSInput(hDrs, 7)
12. msgIn := pmsgMod^.V1
13. /\* Validate input parameters \*/
14. if msgIn.pNC = null
15. or msgIn.pNC^ = ""
16. or (msgIn.pszSourceDRA = null
17. and msgIn.uuidSourceDRA = null)
18. or (DRS\_UPDATE\_ADDRESS in msgIn.ulModifyFields
19. and (msgIn.pszSourceDRA = null
20. or msgIn.pszSourceDRA = ""))
21. or (DRS\_UPDATE\_SCHEDULE in msgIn.ulModifyFields
22. and msgIn.rtSchedule = null)
23. or msgIn.ulModifyFields = 0
24. or msgIn.ulModifyFields -
25. {DRS\_UPDATE\_ADDRESS, DRS\_UPDATE\_SCHEDULE, DRS\_UPDATE\_FLAGS}
26. ≠ {}
27. or msgIn.ulOptions – {DRS\_ASYNC\_OP} ≠ {} then
28. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
29. endif
30. /\* Validate the specified NC \*/
31. options := msgIn.ulOptions
32. nc := msgIn.pNC^
33. if not ObjExists(nc) then
34. return ERROR\_DS\_DRA\_BAD\_NC
35. endif
36. if not AccessCheckCAR(nc, DS-Replication-Manage-Topology) then
37. return ERROR\_DS\_DRA\_ACCESS\_DENIED
38. endif
39. if DRS\_ASYNC\_OP in options then
40. Asynchronous Processing: Initiate a logical thread of control
41. to process the remainder of this request asynchronously
42. return 0
43. endif
44. /\* Find the specified repsFrom. \*/
45. if (msgIn.uuidSourceDRA ≠ null ) then
46. rf := select one v from nc!repsFrom
47. where (v.uuidDsa = msgIn.uuidSourceDRA)
48. else
49. rf := select one v from nc!repsFrom
50. where (v.serverAddress = msgIn.pszSourceDRA)
51. end if
52. if rf = null then
53. return ERROR\_DS\_DRA\_NO\_REPLICA
54. endif
55. /\* Update the specified repsFrom. \*/
56. nc!repsFrom := nc!repsFrom - {rf}
57. if DRS\_UPDATE\_ADDRESS in msgIn.ulModifyFields then
58. rf.serverAddress := msgIn.pszSourceDRA
59. endif
60. if DRS\_UPDATE\_SCHEDULE in msgIn.ulModifyFields then
61. rf.schedule := msgIn.rtSchedule
62. endif
63. if DRS\_UPDATE\_FLAGS in msgIn.ulModifyFields then
64. rf.replicaFlags := msgIn.ulReplicaFlags
65. endif
66. nc!repsFrom := nc!repsFrom + {rf}
67. return 0

### IDL\_DRSReplicaSync (Opnum 2)

The IDL\_DRSReplicaSync method triggers [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) from another [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

1. ULONG IDL\_DRSReplicaSync(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwVersion,
4. [in, ref, switch\_is(dwVersion)]
5. DRS\_MSG\_REPSYNC\* pmsgSync
6. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwVersion:** The version of the request message.

**pmsgSync:** A pointer to the request message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_REPSYNC

The DRS\_MSG\_REPSYNC union defines the request messages sent to the [IDL\_DRSReplicaSync](#Section_25c71d91051f4c26977fa70892f29b00) method. Only one version, identified by *dwVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_REPSYNC\_V1 V1;
6. [case(2)]
7. DRS\_MSG\_REPSYNC\_V2 V2;
8. } DRS\_MSG\_REPSYNC;

**V1:**  The version 1 request.

**V2**: The version 2 request.

##### DRS\_MSG\_REPSYNC\_V1

The DRS\_MSG\_REPSYNC\_V1 structure defines a request message sent to the [IDL\_DRSReplicaSync](#Section_25c71d91051f4c26977fa70892f29b00) method.

1. typedef struct {
2. [ref] DSNAME\* pNC;
3. UUID uuidDsaSrc;
4. [unique] [string] char\* pszDsaSrc;
5. ULONG ulOptions;
6. } DRS\_MSG\_REPSYNC\_V1;

**pNC:**  A pointer to [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the root of an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) on the server.

**uuidDsaSrc:**  The [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1).

**pszDsaSrc:**  The transport-specific [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**ulOptions:**  The [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) flags.

##### DRS\_MSG\_REPSYNC\_V2

The DRS\_MSG\_REPSYNC\_V2 structure defines a request message sent to the [IDL\_DRSReplicaSync](#Section_25c71d91051f4c26977fa70892f29b00) method.

1. typedef struct {
2. [ref] DSNAME\* pNC;
3. UUID uuidDsaSrc;
4. [unique] [string] char\* pszDsaSrc;
5. ULONG ulOptions;
6. GUID correlationID;
7. [unique] VAR\_SIZE\_BUFFER\_WITH\_VERSION\* pReservedBuffer;
8. } DRS\_MSG\_REPSYNC\_V2;

**pNC:**  A pointer to [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the root of an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) on the server.

**uuidDsaSrc:**  The [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1).

**pszDsaSrc:**  The transport-specific [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**ulOptions:**  The [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) flags.

**correlationID**: An identifier for the operation that the DC can use for implementation-defined troubleshooting. There are no normative constraints on this value, nor does the value figure in any normative processing rules.

**pReservedBuffer**: A pointer to a VAR\_SIZE\_BUFFER\_WITH\_VERSION structure (section [5.219](#Section_589574c1eaa1456fac53de597b2cff6b)). MUST be a null pointer.

#### Server Behavior of the IDL\_DRSReplicaSync Method

*Informative summary of behavior*: The server starts or resumes a [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16) by sending an IDL\_DRSGetNCChanges request to the specified [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). If ulOptions contains DRS\_ASYNC\_OP, the server performs this operation asynchronously.

1. ULONG
2. IDL\_DRSReplicaSync(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwVersion,
5. [in, ref, switch\_is(dwVersion)]
6. DRS\_MSG\_REPSYNC \*pmsgSync);
7. options: DRS\_OPTIONS
8. nc: DSName
9. rf: sequence of RepsFrom
10. msgIn: DRS\_MSG\_REPSYNC\_V1
11. err: ULONG
12. ValidateDRSInput(hDrs, 2)
13. /\* Validate the version \*/
14. if dwVersion ≠ 1 then
15. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
16. endif
17. msgIn := pmsgSync^.V1
18. /\* Validate input params \*/
19. options := msgIn.ulOptions
20. if msgIn.pNC = null
21. or (not DRS\_SYNC\_ALL in options /\* See product behavior note below.\*/
22. and msgIn.uuidDsaSrc = null
23. and msgIn.pszDsaSrc = null) then
24. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
25. endif
26. /\* Validate the specified NC. \*/
27. nc := msgIn.pNC^
28. if not ObjExists(nc) then
29. return ERROR\_DS\_DRA\_BAD\_NC
30. endif
31. if (DRS\_SYNC\_BYNAME in options and msgIn.pszDsaSrc = null)
32. or (not DRS\_SYNC\_BYNAME in options and msgIn.uuidDsaSrc = null)
33. or (not DRS\_SYNC\_BYNAME in options and msgIn.uuidDsaSrc = NULLGUID) then
34. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
35. endif
36. if AccessCheckCAR(nc, DS-Replication-Synchronize) then
37. return ERROR\_DS\_DRA\_ACCESS\_DENIED
38. endif
39. if DRS\_ASYNC\_OP in options then
40. Asynchronous Processing: Initiate a logical thread of control
41. to process the remainder of this request asynchronously
42. return 0
43. endif
44. rf := select all v in nc!repsFrom
45. where DRS\_SYNC\_ALL in options
46. or (DRS\_SYNC\_BYNAME in options
47. and v.naDsa = msgIn.pszDsaSrc)
48. or (not DRS\_SYNC\_BYNAME in options
49. and v.uuidDsa = msgIn.uuidDsaSrc)
50. if rf = null then
51. return ERROR\_DS\_DRA\_NO\_REPLICA
52. endif
53. foreach r in rf
54. msgRequest: DRS\_MSG\_GETCHGREQ
55. cMaxObjects: ULONG
56. cMaxBytes: ULONG
57. versionRequestMsg: DWORD
58. outVersion: DWORD
59. msgOut: DRS\_MSG\_GETCHGREPLY
61. versionRequestMsg := The version number of the input message negotiated between the
62. client and server (section 4.1.10.4.1).
63. cMaxObjects := Implementation-specific value.
64. cMaxBytes := Implementation-specific value.
65. if DRS\_UPDATE\_NOTIFICATION in options
66. and not DRS\_TWOWAY\_SYNC in options
67. and DRS\_NEVER\_NOTIFY in r.V2.ulReplicaFlags then
68. return ERROR\_DS\_DRA\_NO\_REPLICA
69. endif
70. /\* Replicate nc from the DC specified by r.uuidDsa. \*/
72. ReplicateNCRequestMsg(
73. hDrs,
74. versionRequestMsg,
75. nc,
76. r,
77. options,
78. cMaxObjects,
79. cMaxBytes,
80. ADDR(msgRequest))
81. err := IDL\_DRSGetNCChanges(
82. hDrsSrc,
83. versionRequestMsg,
84. msgRequest,
85. ADDR(outVersion),
86. ADDR(msgOut))
87. if err = 0
88. and not DRS\_MAIL\_REP in msgIn.ulOptions
89. then
90. Wait for the response, process it (section 4.1.10.6), send the next request,
91. etc., until the replication cycle is complete.
92. If there are any failures from this replication attempt, assign an
93. appropriate error value to err, and then break out of the for loop.
94. endif
95. endfor
96. return err

For information about Windows support for the DRS\_SYNC\_ALL flag, see the product behavior note in section [5.41](#Section_AC9C8A11CD464080ACBF9FAA86344030).

### IDL\_DRSReplicaVerifyObjects (Opnum 22)

The IDL\_DRSReplicaVerifyObjects method verifies the existence of [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) by comparing against a [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) of the same [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) on a reference [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), optionally deleting any objects that do not exist on the reference DC.

1. ULONG IDL\_DRSReplicaVerifyObjects(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwVersion,
4. [in, ref, switch\_is(dwVersion)]
5. DRS\_MSG\_REPVERIFYOBJ\* pmsgVerify
6. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwVersion:** The version of the request message.

**pmsgVerify:** A pointer to the request message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_REPVERIFYOBJ

The DRS\_MSG\_REPVERIFYOBJ union defines the request messages sent to the [IDL\_DRSReplicaVerifyObjects](#Section_8dba150d50f647f1941e1a606c30db0b) method. Only one version, identified by *dwVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_REPVERIFYOBJ\_V1 V1;
6. } DRS\_MSG\_REPVERIFYOBJ;

**V1:**  The version 1 request.

##### DRS\_MSG\_REPVERIFYOBJ\_V1

The DRS\_MSG\_REPVERIFYOBJ\_V1 structure defines a request message sent to the [IDL\_DRSReplicaVerifyObjects](#Section_8dba150d50f647f1941e1a606c30db0b) method.

1. typedef struct {
2. [ref] DSNAME\* pNC;
3. UUID uuidDsaSrc;
4. ULONG ulOptions;
5. } DRS\_MSG\_REPVERIFYOBJ\_V1;

**pNC:**  The [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) to verify.

**uuidDsaSrc:**  The objectGUID of the [**nTDSDSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8) for the reference [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**ulOptions:**  0 to [**expunge**](#gt_c947d085-898e-44c0-a849-47c3b817b7b7) each [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that is not verified, or 1 to log an event that identifies each such object.

#### Method-Specific Abstract Types and Procedures

##### GetRemoteUTD

1. procedure GetRemoteUTD(
2. dsa: DSName,
3. nc: DSName,
4. var uTDVec: UPTODATE\_VECTOR\_V1\_EXT
5. ): ULONG

The GetRemoteUTD procedure uses the [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47) method to remotely retrieve the [UPTODATE\_VECTOR\_V1\_EXT](#Section_462b424ab50a4c4aa81f48d0f4cf40fe) for the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) with the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) *nc* from the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) whose nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) has the DSName *dsa*. The procedure returns either an implementation-specific value from the client implementation of the IDL\_DRSGetReplInfo method, or the value returned by the remote server's IDL\_DRSGetReplInfo method.

##### ObjectExistsAtDC

1. procedure ObjectExistsAtDC(o: DSName, dsa: DSName): boolean

The ObjectExistsAtDC procedure checks that the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *o* exists on the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) whose nTDSDSA object has the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) *dsa* by verifying that the DC holds an object *o'* whose [**objectGUID**](#gt_ad613dff-e9c4-4cb6-ad6b-0ce52038ceb5) value is equal to that of object *o*. If the object exists, the procedure returns true; otherwise, the procedure returns false.

#### Server Behavior of the IDL\_DRSReplicaVerifyObjects Method

*Informative summary of behavior*: Let N be the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) pNC^, and let the reference [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) be the DC corresponding to the nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) uuidDsaSrc.

For the purposes of this method, an object *exists* within an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) if it is either an object or a [**tombstone**](#gt_9d8e0963-13fa-4e19-a97f-7ce6bc90d20f).

Let S be the set of objects that exists in N at the server running IDL\_DRSReplicaVerifyObjects at the time IDL\_DRSReplicaVerifyObjects begins processing. Let the set S' be S minus the members of S that have never existed in N at the reference DC when IDL\_DRSReplicaVerifyObjects begins processing. The members of (S - S') are objects recently added to N on the server, since otherwise they would have replicated to the reference DC. The set S' is computable using the replUpToDateVector for N at the server and at the reference DC.

For each object *o* in S' that does not exist in N at the reference DC while IDL\_DRSReplicaVerifyObjects is processing, either [expunge](#Section_b4dfb24533e244eba5e16419d74907ca) *o* at the server (if ulOptions = 0) or log an administrator-visible event at the server (if ulOptions = 1).

If an object goes out of existence in N at the reference DC during processing of IDL\_DRSReplicaVerifyObjects, then there is no requirement on whether IDL\_DRSReplicaVerifyObjects should or should not expunge or log the object at the server.

1. ULONG IDL\_DRSReplicaVerifyObjects(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwVersion,
4. [in, ref, switch\_is(dwVersion)]
5. DRS\_MSG\_REPVERIFYOBJ \*pmsgVerify)
6. err: ULONG
7. msgIn: DRS\_MSG\_REPVERIFYOBJ\_V1
8. nc, refDsa, o: DSName
9. uTDServer, uTDRef, uTDMerge: UPTODATE\_VECTOR\_V1\_EXT
10. sPrime: set of DSName
11. ValidateDRSInput(hDrs, 22)
12. /\* Perform input validation and access check \*/
13. if dwVersion ≠ 0x1 then
14. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
15. endif
16. msgIn := pmsgVerify^.V1
17. if msgIn.pNC = null or
18. msgIn.uuidDsaSrc = NULLGUID then
19. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
20. endif
21. nc := msgIn.pNC^
22. if not FullReplicaExists(nc) and
23. not PartialGCReplicaExists(nc) then
24. return ERROR\_DS\_DRA\_BAD\_NC
25. endif
26. if not AccessCheckCAR(nc, DS-Replication-Manage-Topology) then
27. return ERROR\_DS\_DRA\_ACCESS\_DENIED
28. endif
29. refDsa := select one object o from subtree ConfigNC() where
30. o!objectGUID = msgIn.uuidDsaSrc and nTDSDSA in o!objectClass
31. if refDsa = null then
32. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
33. endif
34. /\* Compute the set S' \*/
35. uTDServer := nc!replUpToDateVector
36. err := GetRemoteUTD(refDsa, nc, uTDRef)
37. if err ≠ 0 then
38. return err
39. endif
40. uTDMerge := MergeUTD(uTDServer, uTDRef)
41. sPrime := select all objects o from subtree-ts-included nc where
42. StampLessThanOrEqualUTD(AttrStamp(o, whenCreated), uTDMerge)
43. /\* Process the set S' \*/
44. for each o in sPrime
45. if not ObjectExistsAtDC(o, refDSA) then
46. if msgIn.ulOptions = 0 then
47. Expunge(o)
48. else if msgIn.ulOptions = 1 then
49. Log a message: o exists on server but does not exist on refDsa
50. endif
51. endfor
52. return 0

Windows behavior about the for loop is specified in the following citation:[<37>](#Appendix_A_37" \o "Product behavior note 37)

#### Examples of the IDL\_DRSReplicaVerifyObjects Method

A client that has bound to DC1 is removing all [**lingering objects**](#gt_79ffc6ee-a12c-4682-970b-409e4b19a23d) on this [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) server with respect to DC2.

##### Initial State

A client has bound to DC1.CONTOSO.COM using the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method and received a [DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1.

Consider the following [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) under the Users [**container**](#gt_c3143e71-2ada-417e-83f4-3ef10eff2c56), "CN=Users,DC=CONTOSO,DC=COM", listed by their [DSName](#Section_a0d5477a522946b9890a54b924d487d1):

| Users at DC1 | Users at DC2 | Notes |
| --- | --- | --- |
| <GUID=f5ef2f4b-a3db-464c-8403-b27aa00b0d5d>;<SID=S-1-5-21-1583212203-607051668-819563750-1107>;CN=Kim Akers, CN=Users,DC=CONTOSO,DC=COM | <GUID=f5ef2f4b-a3db-464c-8403-b27aa00b0d5d>;<SID=S-1-5-21-1583212203-607051668-819563750-1107>;CN=Kim Akers, CN=Users,DC=CONTOSO,DC=COM | Objects are identical. |
| <GUID=89430510-48eb-4e68-aeb1-98a9471f1938>;<SID=S-1-5-21-1583212203-607051668-819563750-1111>; CN=Josh Bailey,CN=Users,DC=CONTOSO,DC=COM |  | "Josh Bailey" was created on DC1 and has not been replicated to DC2 yet. |
| <GUID=833a118e-035f-4702-b67e-9e7c1ada2f57>;<SID=S-1-5-21-1583212203-607051668-819563750-1108>;CN= Eva Corets,CN=Users,DC=CONTOSO,DC=COM |  | "Eva Corets" is a [**lingering object**](#gt_79ffc6ee-a12c-4682-970b-409e4b19a23d) on DC1. |
| <GUID=3cb4b6cf-f220-472a-bd2f-5f1399232ca6>;<SID=S-1-5-21-1583212203-607051668-819563750-1109>;CN= Jim Daly,CN=Users,DC=CONTOSO,DC=COM | <GUID=3cb4b6cf-f220-472a-bd2f-5f1399232ca6>;<SID=S-1-5-21-1583212203-607051668-819563750-1109>;CN= Jim Daly,CN=Users,DC=CONTOSO,DC=COM | The mail [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of "Jim Daly" has been modified on DC1 but this change has not replicated to DC2 yet. |
|  | <GUID=46c1b351-da31-49f2-8437-8d82df024972>;<SID=S-1-5-21-1583212203-607051668-819563750-1604>; CN=Ebru Ersan,CN=Users,DC=CONTOSO,DC=COM | "Ebru Ersan" was created on DC2 and has not been replicated to DC1 yet. |
|  | <GUID=8df1f9bb-7551-46c3-b9c2-c905e9463542>;<SID=S-1-5-21-1583212203-607051668-819563750-1110>; CN= Kari Furse,CN=Users,DC=CONTOSO,DC=COM | "Kari Furse" is a lingering object on DC2. |

Relevant entries of the DS\_REPL\_ATTR\_META\_DATA structure for each object listed above are also captured below to further demonstrate the differences between DC1 and DC2.

Relevant metadata entries for "CN=Kim Akers,CN=Users,DC=CONTOSO,DC=COM" at DC1:

| usnLocalChange | uuidLastOriginatingDsaInvocationID | usnOriginatingChange | ftimeLastOriginatingChange | dwVersion | pszAttributeName |
| --- | --- | --- | --- | --- | --- |
| 13964 | 4875e25f-11a9-4c70-abf4-5fb39529f84b | 13964 | 5/21/2010 18:08:30 | 1 | whenCreated |

Relevant metadata entries for "CN=Josh Bailey,CN=Users,DC=CONTOSO,DC=COM" at DC1:

| usnLocalChange | uuidLastOriginatingDsaInvocationID | usnOriginatingChange | ftimeLastOriginatingChange | dwVersion | pszAttributeName |
| --- | --- | --- | --- | --- | --- |
| 14112 | 4875e25f-11a9-4c70-abf4-5fb39529f84b | 14112 | 5/21/2010 19:11:09 | 1 | whenCreated |

Relevant metadata entries for "CN=Eva Corets,CN=Users,DC=CONTOSO,DC=COM" at DC1:

| usnLocalChange | uuidLastOriginatingDsaInvocationID | usnOriginatingChange | ftimeLastOriginatingChange | dwVersion | pszAttributeName |
| --- | --- | --- | --- | --- | --- |
| 9071 | 4875e25f-11a9-4c70-abf4-5fb39529f84b | 9071 | 1/15/2009 11:05:42 | 1 | whenCreated |

Relevant metadata entries for "CN=Jim Daly,CN=Users,DC=CONTOSO,DC=COM" at DC1:

| usnLocalChange | uuidLastOriginatingDsaInvocationID | usnOriginatingChange | ftimeLastOriginatingChange | dwVersion | pszAttributeName |
| --- | --- | --- | --- | --- | --- |
| 14085 | 4875e25f-11a9-4c70-abf4-5fb39529f84b | 14085 | 5/21/2010 19:06:32 | 1 | whenCreated |
| 14118 | 4875e25f-11a9-4c70-abf4-5fb39529f84b | 14118 | 5/21/2010 19:12:51 | 1 | mail |

Relevant metadata entries for "CN=Kim Akers,CN=Users,DC=CONTOSO,DC=COM" at DC2:

| usnLocalChange | uuidLastOriginatingDsaInvocationID | usnOriginatingChange | ftimeLastOriginatingChange | dwVersion | pszAttributeName |
| --- | --- | --- | --- | --- | --- |
| 12324 | 4875e25f-11a9-4c70-abf4-5fb39529f84b | 13964 | 5/21/2010 18:08:30 | 1 | whenCreated |

Relevant metadata entries for "CN=Jim Daly,CN=Users,DC=CONTOSO,DC=COM" at DC2:

| usnLocalChange | uuidLastOriginatingDsaInvocationID | usnOriginatingChange | ftimeLastOriginatingChange | dwVersion | pszAttributeName |
| --- | --- | --- | --- | --- | --- |
| 12432 | 4875e25f-11a9-4c70-abf4-5fb39529f84b | 14085 | 5/21/2010 19:06:32 | 1 | whenCreated |

Relevant metadata entries for "CN=Ebru Ersan,CN=Users,DC=CONTOSO,DC=COM" at DC2:

| usnLocalChange | uuidLastOriginatingDsaInvocationID | usnOriginatingChange | ftimeLastOriginatingChange | dwVersion | pszAttributeName |
| --- | --- | --- | --- | --- | --- |
| 12451 | 7526f625-51db-4022-8150-59c0286efd82 | 12451 | 5/21/2010 19:19:14 | 1 | whenCreated |

Relevant metadata entries for "CN=Kari Furse,CN=Users,DC=CONTOSO,DC=COM" at DC2:

| usnLocalChange | uuidLastOriginatingDsaInvocationID | usnOriginatingChange | ftimeLastOriginatingChange | dwVersion | pszAttributeName |
| --- | --- | --- | --- | --- | --- |
| 441 | 4875e25f-11a9-4c70-abf4-5fb39529f84b | 5099 | 11/1/2008  04:29:47 | 1 | whenCreated |

The [UPTODATE\_VECTOR\_V1\_EXT](#Section_462b424ab50a4c4aa81f48d0f4cf40fe) structures on DC1 and DC2 are also needed for the [IDL\_DRSReplicaVerifyObjects](#Section_8dba150d50f647f1941e1a606c30db0b) method:

* On DC1:

**dwVersion:** 1

**dwReserved1:** 0

**cNumCursors:** 2

**dwReserved2:** 0

**rgCursors:** An array of UPTODATE\_CURSOR\_V1:

* + First entry:

**uuidDsa:** 4875e25f-11a9-4c70-abf4-5fb39529f84b

**usnHighPropUpdate:** 14621

* + Second entry:

**uuidDsa:** 7526f625-51db-4022-8150-59c0286efd82

**usnHighPropUpdate:** 12448

* On DC2:

**dwVersion:** 1

**dwReserved1:** 0

**cNumCursors:** 2

**dwReserved2:** 0

**rgCursors:** An array of [UPTODATE\_CURSOR\_V1](#Section_cf88f341fb494cd5b7e26920cbd91f1b):

* + First entry:

**uuidDsa:** 4875e25f-11a9-4c70-abf4-5fb39529f84b

**usnHighPropUpdate:** 14107

* + Second entry:

**uuidDsa:** 7526f625-51db-4022-8150-59c0286efd82

**usnHighPropUpdate:** 12992

* + Finally, also relevant to IDL\_DRSReplicaVerifyObjects is the [**nTDSDSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8) for DC2 as seen on DC1:
    - Dn: CN=NTDS Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=CONTOSO,DC=COM
    - 3> objectClass: top; applicationSettings; nTDSDSA;
    - 1> cn: NTDS Settings;
    - 1> distinguishedName: CN=NTDS Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=CONTOSO,DC=COM;
    - 1> objectGUID: e845e047-3850-4a82-8811-a0b9250863c6;

##### Client Request

A client invokes the [IDL\_DRSReplicaVerifyObjects](#Section_8dba150d50f647f1941e1a606c30db0b) method on DC1 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwVersion*: 1
* *pmsgVerify*:

**pNC:** Pointer to the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) structure for DC=CONTOSO,DC=COM

**uuidDsaSrc:** e845e047-3850-4a82-8811-a0b9250863c6

**ulOptions:** 0

##### Server Response

The server returns a code of 0.

##### Final State

The IDL\_DRSReplicaVerifyObjects method has removed all [**lingering objects**](#gt_79ffc6ee-a12c-4682-970b-409e4b19a23d) on DC1 (but not on DC2). The following table compares the Users [**container**](#gt_c3143e71-2ada-417e-83f4-3ef10eff2c56) on DC1 and DC2 after the IDL\_DRSReplicaVerifyObjects method has been successfully returned.

| Users at DC1 | Users at DC2 | Notes |
| --- | --- | --- |
| <GUID=f5ef2f4b-a3db-464c-8403-b27aa00b0d5d>;<SID=S-1-5-21-1583212203-607051668-819563750-1107>;CN=Kim Akers, CN=Users,DC=CONTOSO,DC=COM | <GUID=f5ef2f4b-a3db-464c-8403-b27aa00b0d5d>;<SID=S-1-5-21-1583212203-607051668-819563750-1107>;CN=Kim Akers, CN=Users,DC=CONTOSO,DC=COM | [**Objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) are identical. |
| GUID=89430510-48eb-4e68-aeb1-98a9471f1938>;<SID=S-1-5-21-1583212203-607051668-819563750-1111>; CN=Josh Bailey,CN=Users,DC=CONTOSO,DC=COM |  | "Josh Bailey" was created on DC1 and has not been replicated to DC2 yet. |
|  |  | "Eva Corets" was a lingering object on DC1 and has been [**expunged**](#gt_c947d085-898e-44c0-a849-47c3b817b7b7). |
| <GUID=3cb4b6cf-f220-472a-bd2f-5f1399232ca6>;<SID=S-1-5-21-1583212203-607051668-819563750-1109>;CN= Jim Daly,CN=Users,DC=CONTOSO,DC=COM | <GUID=3cb4b6cf-f220-472a-bd2f-5f1399232ca6>;<SID=S-1-5-21-1583212203-607051668-819563750-1109>;CN= Jim Daly,CN=Users,DC=CONTOSO,DC=COM | The mail [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of "Jim Daly" has been modified on DC1 but this change has not replicated to DC2 yet. |
|  | <GUID=46c1b351-da31-49f2-8437-8d82df024972>;<SID=S-1-5-21-1583212203-607051668-819563750-1604>; CN=Ebru Ersan,CN=Users,DC=CONTOSO,DC=COM | "Ebru Ersan" was created on DC2 and has not been replicated to DC1 yet. |
|  | <GUID=8df1f9bb-7551-46c3-b9c2-c905e9463542>;<SID=S-1-5-21-1583212203-607051668-819563750-1110>; CN= Kari Furse,CN=Users,DC=CONTOSO,DC=COM | "Kari Furse" is a lingering object on DC2. |

### IDL\_DRSUnbind (Opnum 1)

The IDL\_DRSUnbind method destroys a context handle previously created by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

1. ULONG IDL\_DRSUnbind(
2. [in, out, ref] DRS\_HANDLE\* phDrs
3. );

**phDrs:** A pointer to the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the IDL\_DRSBind method. The value is set to null on return.

**Return Values:** 0 if successful, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if a failure occurs.

**Exceptions Thrown**: This method might throw the following exception beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE.

#### Server Behavior of the IDL\_DRSUnbind Method

*Informative summary of behavior*: The server releases any resources associated with the context handle, making the context handle unusable by the client. The server sets *phDrs* to null.

1. ULONG
2. IDL\_DRSUnbind(
3. [in, out, ref] DRS\_HANDLE \*phDrs)
4. ValidateDRSInput(hDrs, 1)
5. phDrs^ := null
6. return 0

### IDL\_DRSUpdateRefs (Opnum 4)

The IDL\_DRSUpdateRefs method adds or deletes a value from the repsTo of a specified [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

1. ULONG IDL\_DRSUpdateRefs(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwVersion,
4. [in, ref, switch\_is(dwVersion)]
5. DRS\_MSG\_UPDREFS\* pmsgUpdRefs
6. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwVersion:** The version of the request message.

**pmsgUpdRefs:** A pointer to the request message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_UPDREFS

The DRS\_MSG\_UPDREFS union defines the request message versions sent to the [IDL\_DRSUpdateRefs](#Section_a273bbcfaeca46088ad4127d3e597cd4) method. Only one version, identified by *dwVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_UPDREFS\_V1 V1;
6. [case(2)]
7. DRS\_MSG\_UPDREFS\_V2 V2;
8. } DRS\_MSG\_UPDREFS;

**V1:**  The version 1 request.

**V2**: The version 2 request.

##### DRS\_MSG\_UPDREFS\_V1

The DRS\_MSG\_UPDREFS\_V1 structure defines a request message sent to the [IDL\_DRSUpdateRefs](#Section_a273bbcfaeca46088ad4127d3e597cd4) method.

1. typedef struct {
2. [ref] DSNAME\* pNC;
3. [ref] [string] char\* pszDsaDest;
4. UUID uuidDsaObjDest;
5. ULONG ulOptions;
6. } DRS\_MSG\_UPDREFS\_V1;

**pNC:**  A pointer to the [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) of the root of an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) on the server.

**pszDsaDest:**  The transport-specific [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**uuidDsaObjDest:**  The [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1).

**ulOptions:**  The [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) that control the [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493).

##### DRS\_MSG\_UPDREFS\_V2

The DRS\_MSG\_UPDREFS\_V2 structure defines a request message sent to the [IDL\_DRSUpdateRefs](#Section_a273bbcfaeca46088ad4127d3e597cd4) method.

1. typedef struct {
2. [ref] DSNAME\* pNC;
3. [ref] [string] char\* pszDsaDest;
4. UUID uuidDsaObjDest;
5. ULONG ulOptions;
6. GUID correlationID;
7. [unique] VAR\_SIZE\_BUFFER\_WITH\_VERSION\* pReservedBuffer;
8. } DRS\_MSG\_UPDREFS\_V2;

**pNC:**  A pointer to the [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) of the root of an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) on the server.

**pszDsaDest:**  The transport-specific [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**uuidDsaObjDest:**  The [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1).

**ulOptions:**  The [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) that control the [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493).

**correlationID**: An identifier for the operation that the DC can use for implementation-defined troubleshooting. There are no normative constraints on this value, nor does the value figure in any normative processing rules.

**pReservedBuffer**: A pointer to a VAR\_SIZE\_BUFFER\_WITH\_VERSION structure (section [5.219](#Section_589574c1eaa1456fac53de597b2cff6b)). MUST be a null pointer.

#### Server Behavior of the IDL\_DRSUpdateRefs Method

*Informative summary of behavior*: If ulOptions contains DRS\_ADD\_REF, the server adds a value to the repsTo of the specified [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210); if ulOptions contains DRS\_DEL\_REF, the server deletes a value. If these options are combined, the Delete operation is done before the Add operation; if a corresponding value does not already exist, this is the same as if ulOptions contained DRS\_ADD\_REF but not DRS\_DEL\_REF. The client includes DRS\_WRIT\_REP in ulOptions if the specified NC replica is writable. The client specifies both **pszDsaDest** and **uuidDsaObjDest** to identify the value to be added or removed. If ulOptions contains DRS\_ASYNC\_OP, the server processes the request asynchronously. If the server adds a value to repsTo, the value has ulReplicaFlags equal to ulOptions ∩ {DRS\_WRIT\_REP}.

1. ULONG IDL\_DRSUpdateRefs(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwVersion,
4. [in, ref, switch\_is(dwVersion)] DRS\_MSG\_UPDREFS \*pmsgUpdRefs);
5. msgIn: DRS\_MSG\_UPDREFS\_V1
6. options: DRS\_OPTIONS
7. err: DWORD
8. nc: DSName
9. ValidateDRSInput(hDrs, 4)
10. if dwVersion ≠ 1 then
11. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
12. endif
13. msgIn := pmsgUpdRefs^.V1
14. options := msgIn.ulOptions
15. if msgIn.pNC = null or
16. (msgIn.pszDsaDest = null) or
17. (msgIn.uuidDsaObjDest = null) or
18. (options ∩ {DRS\_ADD\_REF, DRS\_DEL\_REF} = null)
19. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
20. endif
21. nc := msgIn.pNC^
22. if (options – {DRS\_ASYNC\_OP, DRS\_GETCHG\_CHECK, DRS\_WRIT\_REP, DRS\_DEL\_REF, DRS\_ADD\_REF, DRS\_REF\_GCSPN} != 0) then
23. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
24. if ((DRS\_WRIT\_REP in options) and not (IT\_WRITE in nc!instanceType)) or
25. not ObjExists(nc) then
26. return ERROR\_DS\_DRA\_BAD\_NC
27. endif
28. if not AccessCheckCAR(nc, DS-Replication-Manage-Topology) then
29. return ERROR\_DS\_DRA\_ACCESS\_DENIED
30. endif
31. /\* Perform repsTo value add, delete, or combination of add/delete to the specified NC replica,
32. \* the result value is a Windows error code or 0
33. result := UpdateRefs(pmsgIn^.V1)
34. if(result ≠ ERROR\_SUCCESS) then
35. return result
36. endif
37. /\* If DRS\_GETCHG\_CHECK is specified, ERROR\_DS\_DRA\_REF\_NOT\_FOUND and
38. \* ERROR\_DS\_DRA\_REF\_ALREADY\_EXISTS are ignored. \*/
39. if DRS\_GETCHG\_CHECK in options and
40. (err = ERROR\_DS\_DRA\_REF\_NOT\_FOUND or err = ERROR\_DS\_DRA\_REF\_ALREADY\_EXISTS) then
41. err := 0
42. endif
43. return err

#### Examples of the IDL\_DRSUpdateRefs Method

##### Adding a repsTo Entry

This example shows how to add a new **repsTo** entry by calling [IDL\_DRSUpdateRefs (section 4.1.26)](#Section_a273bbcfaeca46088ad4127d3e597cd4) with the *DRS\_ADD\_REF* parameter.

###### Initial State

The repsTo [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) CONTOSO.COM on DC1 does not contain a value:

ldap\_search\_s("DC=CONTOSO,DC=COM", *baseObject*, "(objectclass=\*)", [*repsTo*])

Result <0>: (null)

Matched DNs:

Getting 1 entry:

>> Dn: DC=CONTOSO,DC=COM

###### Client Request

DC2 invokes the IDL\_DRSUpdateRefs method against DC1, with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwVersion* = 1
* *pmsgUpdRefs* = 0x0006fe08 ; Pointer to the following structure:
* pNC: Pointer to the DSNAME structure for DC=CONTOSO,DC=COM
* pszDsaDest: "5fe84f18-3765-4ca3-b895-47802a7ab74f.\_msdcs.CONTOSO.COM"
* uuidDsaObjDest: 5fe84f18-3765-4ca3-b895-47802a7ab74f
* ulOptions: DRS\_WRIT\_REP | DRS\_ADD\_REF

###### Server Response

Return code of 0.

###### Final State

The repsTo [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) CONTOSO.COM on DC1 contains one value:

ldap\_search\_s("DC=CONTOSO,DC=COM", *baseObject*, "(objectclass=\*)", [*repsTo*])

Result <0>: (null)

Matched DNs:

Getting 1 entry:

>> Dn: DC=CONTOSO,DC=COM

1> repsTo: dwVersion = 2,

* V2.cb: 592, V2.cConsecutiveFailures: 0, V2.timeLastSuccess: 12924245513,
* V2.timeLastAttempt: 0, V2.ulResultLastAttempt: 0,
* V2.cbOtherDraOffset: 216,
* V2.cbOtherDra: 376, V2.ulReplicaFlags: 16, V2.rtSchedule: <ldp:skipped>,
* V2.usnvec.usnHighObjUpdate: 0, V2.usnvec.usnHighPropUpdate: 0,
* V2.uuidDsaObj: 5fe84f18-3765-4ca3-b895-47802a7ab74f
* V2.uuidInvocId: 00000000-0000-0000-0000-000000000000
* V2.uuidTransportObj: 00000000-0000-0000-0000-000000000000
* V2.cbPASDataOffset: 0
* V2~PasData: (none)
* v2~pdsa\_rpc\_inst
* v2.pszDSIServer 5fe84f18-3765-4ca3-b895-47802a7ab74f.\_msdcs.CONTOSO.COM
* v2.pszDSIAnnotation (null)
* v2.pszDSIInstance 5fe84f18-3765-4ca3-b895-47802a7ab74f.\_msdcs.CONTOSO.COM
* v2.pguidDSIInstance (null);

##### Replacing a repsTo Entry

This example shows how to semantically [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) an existing **repsTo** entry by calling [IDL\_DRSUpdateRefs](#Section_a273bbcfaeca46088ad4127d3e597cd4) with the *DRS\_ADD\_REF* and *DRS\_DEL\_REF* parameters.

###### Initial State

The ldap search

ldap\_search\_s(ld, "DC=CONTOSO,DC=com", 0, "(objectclass=\*)",[repsTo])

returns

Getting 1 entry:

>> Dn: DC=CONTOSO,DC=COM

repsTo (2): dwVersion = 2

* v2.cb: 592, v2.cConsecutive Failures: 0, v2.timeLastSuccess: 12924315918,
* V2.timeLastAttempt: 12924315918, V2.ulResultLastAttempt:0,
* V2.cbOtherDraOffset: 216,
* V2.cbOtherDra: 376, V2.ulReplicaFlags: 16, V2.rtSchedule: <ldp:skipped>,
* V2.usnvec.usnHighObjUpdate: 0, v2.usnvec.usnHighPropUpdate:0
* V2.pszUuidDsaObj: 5fe84f18-3765-4ca3-b895-47802a7ab74f
* V2.pszUuidInvocId: 00000000-0000-0000-0000-000000000000
* V2.pszUuidTransportObj: 00000000-0000-0000-0000-000000000000
* V2.cbPASDataOffset: 0 v2~PasData: (none)
* V2~pdsa\_rpc\_inst
* V2.pszDSIServer 5fe84f18-3765-4ca3-b895-47802a7ab74f.\_msdcs.CONTOSO.COM
* V2.pszDSIAnnotation (null)
* V2.pszDSIInstance 5fe84f18-3765-4ca3-b895-47802a7ab74f.\_msdcs.CONTOSO.COM
* V2.pguidDSIInstance (null)

###### Client Request

A client invokes the [IDL\_DRSUpdateRefs (section 4.1.26)](#Section_a273bbcfaeca46088ad4127d3e597cd4) method against DC1 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted).

* *dwVersion* = 1
* *pmsgUpdRefs* = 0x0006fe08 ; Pointer to the following structure:
  + pNC: Pointer to the DSNAME structure for DC=CONTOSO,DC=COM
  + pszDsaDest : "5fe84f18-3765-4ca3-b895-47802a7ab74f.\_msdcs.contoso.com"
  + uuidDsaObjDest: \_GUID { 5fe84f18-3765-4ca3-b895-47802a7ab74f }
  + ulOptions: DRS\_WRIT\_REP | DRS\_DEL\_REF | DRS\_ADD\_REF

###### Server Response

Return code of 0.

###### Final State

The ldap search

ldap\_search\_s(ld, "DC=CONTOSO,DC=com", 0, "(objectclass=\*)",[repsTo])

returns

Getting 1 entry:

>> Dn: DC=CONTOSO,DC=COM

repsTo (2): dwVersion = 2,

* v2.cb: 592, v2.cConsecutive Failures: 0, v2.timeLastSuccess: 12924320155
* V2.timeLastAttempt: 0, V2.ulResultLastAttempt: 0,
* V2.cbOtherDraOffset: 216,
* V2.cbOtherDra: 376, V2.ulReplicaFlags: 16, V2.rtSchedule: <ldp:skipped>,
* V2.usnvec.usnHighObjUpdate: 0, v2.usnvec.usnHighPropUpdate:0
* V2.pszUuidDsaObj: 5fe84f18-3765-4ca3-b895-47802a7ab74f
* V2.pszUuidInvocId: 00000000-0000-0000-0000-000000000000
* V2.pszUuidTransportObj: 00000000-0000-0000-0000-000000000000
* V2.cbPASDataOffset: 0 v2~PasData: (none)
* v2~pdsa\_rpc\_inst
* v2.pszDSIServer 5fe84f18-3765-4ca3-b895-47802a7ab74f.\_msdcs.CONTOSO.COM
* v2.pszDSIAnnotation (null)
* v2.pszDSIInstance 5fe84f18-3765-4ca3-b895-47802a7ab74f.\_msdcs.CONTOSO.COM
* v2.pguidDSIInstance (null);

### IDL\_DRSVerifyNames (Opnum 8)

The IDL\_DRSVerifyNames method resolves a sequence of [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) identities.

1. ULONG IDL\_DRSVerifyNames(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_VERIFYREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_VERIFYREPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_VERIFYREQ

The DRS\_MSG\_VERIFYREQ union defines the request messages sent to the [IDL\_DRSVerifyNames](#Section_80739a29e8ed44788490475a18e9779d) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_VERIFYREQ\_V1 V1;
6. } DRS\_MSG\_VERIFYREQ;

**V1:**  The version 1 request.

##### DRS\_MSG\_VERIFYREQ\_V1

The DRS\_MSG\_VERIFYREQ\_V1 structure defines a request message sent to the [IDL\_DRSVerifyNames](#Section_80739a29e8ed44788490475a18e9779d) method.

1. typedef struct {
2. DWORD dwFlags;
3. [range(1,10000)] DWORD cNames;
4. [size\_is(cNames)] DSNAME\*\* rpNames;
5. ATTRBLOCK RequiredAttrs;
6. SCHEMA\_PREFIX\_TABLE PrefixTable;
7. } DRS\_MSG\_VERIFYREQ\_V1;

**dwFlags:**  The type of name to be verified; MUST have one of the following values:

| Value | Meaning |
| --- | --- |
| DRS\_VERIFY\_DSNAMES  0x00000000 | Verify DSName values. |
| DRS\_VERIFY\_SIDS  0x00000001 | Verify objectSid values. |
| DRS\_VERIFY\_SAM\_ACCOUNT\_NAMES  0x00000002 | Verify sAMAccountName values. |
| DRS\_VERIFY\_FPOS  0x00000003 | Verify foreign [**principal**](#gt_8492780e-99e2-47ba-8553-aedb8de9f9c0) [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) names. |

**cNames:**  The number of items in the **rpNames** array.

**rpNames:**  An array of pointers to [DSNames](#Section_a0d5477a522946b9890a54b924d487d1) that need to be verified.

**RequiredAttrs:**  The list of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) to be retrieved for each name that is verified.

**PrefixTable:**  The [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) used to translate [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values in **RequiredAttrs** to [OID](#Section_339504853a964b668a28a3a33e80302b) values.

##### DRS\_MSG\_VERIFYREPLY

The DRS\_MSG\_VERIFYREPLY union defines the response messages received from the [IDL\_DRSVerifyNames](#Section_80739a29e8ed44788490475a18e9779d) method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_VERIFYREPLY\_V1 V1;
6. } DRS\_MSG\_VERIFYREPLY;

**V1:**  The version 1 reply.

##### DRS\_MSG\_VERIFYREPLY\_V1

The DRS\_MSG\_VERIFYREPLY\_V1 structure defines a response message received from the [IDL\_DRSVerifyNames](#Section_80739a29e8ed44788490475a18e9779d) method.

1. typedef struct {
2. DWORD error;
3. [range(0,10000)] DWORD cNames;
4. [size\_is(cNames)] ENTINF\* rpEntInf;
5. SCHEMA\_PREFIX\_TABLE PrefixTable;
6. } DRS\_MSG\_VERIFYREPLY\_V1;

**error:**  Unused. MUST be 0 and ignored.

**cNames:**  The number of items in the **rpEntInf** array.

**rpEntInf:**  An array of [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b) structures that contain the [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) requested in the **RequiredAttrs** field of the input [DRS\_MSG\_VERIFYREQ\_V1](#Section_4593a76b71f14e4cb5e14d2e27afb3cb) structure if the corresponding name is verified.

**PrefixTable:**  The [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) used to translate [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values in the response to [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

#### Server Behavior of the IDL\_DRSVerifyNames Method

*Informative summary of behavior*: The server resolves each of a sequence of [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) names and returns its [**DSName**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a) and the values of zero or more of its [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f). The type of the input object name is indicated by the **dwFlags** field in the request. The [IDL\_DRSVerifyNames](#Section_80739a29e8ed44788490475a18e9779d) method verifies the names of both deleted and normal objects.

1. ULONG
2. IDL\_DRSVerifyNames(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_VERIFYREQ \*pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DRS\_MSG\_VERIFYREPLY \*pmsgOut);
10. msgIn: DRS\_MSG\_VERIFYREQ\_V1
11. msgOut: DRS\_MSG\_VERIFYREPLY\_V1
12. nc, d: DSName
13. o: sequence of DSName
14. i, j, k: int
15. domainName, username: unicodestring
16. done: boolean
17. attribute: ATTRTYP
18. FilterPAS: PARTIAL\_ATTR\_VECTOR\_V1\_EXT
19. GCPas: PARTIAL\_ATTR\_VECTOR\_V1\_EXT
20. referredDomain: unicodestring
21. ValidateDRSInput(hDrs, 8)
22. pdwOutVersion^ := 1
23. pmsgOut^.V1.error := 0
24. pmsgOut^.V1.cNames := 0
25. pmsgOut^.V1.rpEntInf := null
26. pmsgOut^.V1.PrefixTable.PrefixCount := 0
27. pmsgOut^.V1.PrefixTable.pPrefixEntry := null
28. /\* Perform input validation and access check \*/
29. if dwInVersion ≠ 0x1 then
30. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
31. endif
32. msgIn := pmsgIn^.V1
33. if msgIn.dwFlags ≠ DRS\_VERIFY\_DSNAMES and
34. msgIn.dwFlags ≠ DRS\_VERIFY\_SAM\_ACCOUNT\_NAMES and
35. msgIn.dwFlags ≠ DRS\_VERIFY\_SIDS and
36. msgIn.dwFlags ≠ DRS\_VERIFY\_FPOS then
37. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
38. endif
39. if msgIn.cNames > 0 and msgIn.rpNames = null then
40. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
41. endif
42. if (msgIn.dwFlags = DRS\_VERIFY\_SIDS or
43. msgIn.dwFlags = DRS\_VERIFY\_SAM\_ACCOUNT\_NAMES or
44. msgIn.dwFlags = DRS\_VERIFY\_FPOS) and
45. not IsGC() then
46. return ERROR\_DS\_GC\_REQUIRED
47. endif
48. if msgIn.dwFlags = DRS\_VERIFY\_DSNAMES and not IsGC() then
49. for i := 0 to msgIn.cNames-1
50. if DefaultNC() ≠ GetObjectNC(msgIn.rpNames[i]^) then
51. return ERROR\_DS\_GC\_REQUIRED
52. endif
53. endfor
54. endif
55. /\* Compute output \*/
56. msgOut.PrefixTable := dc.prefixTable
57. for i := 0 to msgIn.cNames - 1
58. d := msgIn.rpNames[i]
59. o := null
60. done := false
61. if msgIn.dwFlags = DRS\_VERIFY\_SAM\_ACCOUNT\_NAMES then
62. domainName := DomainNameFromNT4AccountName(d.dn)
63. username := UserNameFromNT4AccountName(d.dn)
64. if domainName ≠ null and username ≠ null and
65. IsDomainNameInTrustedForest(domainName, referredDomain) then
66. /\* Provide a hint as to which forest this name could be coming
67. \* from. Note that 0xFFFF0009 is a hardcoded attribute ID
68. \* recognized by clients of this method. This attribute ID does
69. \* not correspond to any attribute defined in the schema. \*/
70. msgOut.rpEntInf[i].pName := null
71. msgOut.rpEntInf[i].AttrBlock.AttrCount := 1
72. msgOut.rpEntInt[i].AttrBlock.pAttr[0].AttrTyp := 0xFFFF0009
73. msgOut.rpEntInf[i].AttrBlock.pAttr[0].AttrVal.valCount := 1
74. msgOut.rpEntInf[i].AttrBlock.pAttr[0].AttrVal.pAVal[0].valLen
75. := Length in characters of domainName, excluding any
76. terminating null
77. msgOut.rpEntInf[i].AttrBlock.pAttr[0].AttrVal.pAVal[0].pAVal :=
78. referredDomain
79. done := true
80. endif
81. endif
82. if not done
83. /\* locate object or objects in question \*/
84. if msgIn.dwFlags = DRS\_VERIFY\_DSNAMES then
85. if ObjExists(d) then
86. o := {d}
87. endif
88. else if msgIn.dwFlags = DRS\_VERIFY\_SIDS then
89. o := select all v from all-ts-included
90. where v!objectSid = d.sid and
91. foreignSecurityPrincipal not in v!objectClass
92. else if msgIn.dwFlags = DRS\_VERIFY\_SAM\_ACCOUNT\_NAMES then
93. if domainName ≠ null and username ≠ null then
94. nc := select one v from all
95. where v!nETBIOSName = domainName and GetObjectNC(v)= v
96. /\* The following query returns both normal objects
97. and tombstones \*/
98. o := select all v from subtree-ts-included nc where
99. v!sAMAccountName = username
100. else
101. o := select all v from all-ts-included
102. where v!userPrincipalName =
103. d.dn
104. endif
105. else if msgIn.dwFlags = DRS\_VERIFY\_FPOS then
106. o := select all v from all-ts-included
107. where v!objectSid = d.sid
108. and foreignSecurityPrincipal in v!objectClass
109. endif
110. /\* Compute returned info and get requested attributes \*/
111. if o.length = 1 and AccessCheckCAR(GetObjectNC(o[0]), DS-Replication-Get-Changes) then
112. msgOut.rpEntInf[i].pName = o[0]!distinguishedName
113. if MasterReplicaExists(GetObjectNC(o[0])) then
114. msgOut.rpEntInf[i].ulFlags := ENTINF\_FROM\_MASTER
115. else
116. msgOut.rpEntEnf[i].ulFlags := 0
117. endif
118. msgOut.rpEntInf[i].AttrBlock.AttrCount :=
119. msgIn.RequiredAttrs.AttrCount
120. FilterPas := FilteredPAS()
121. GCPas := GCPAS()
122. for j := 0 to msgIn.RequiredAttrs.AttrCount - 1
123. if AmILHServer() then
124. if (not (msgIn.RequiredAttrs.pAttr[j].AttrType in FilterPas
125. &&
126. msgIn.RequiredAttrs.pAttr[j].AttrType in GCPas))
127. then
128. /\* skip requested attributes not part of both FilterPAS
129. and GCPas \*/
130. msgOut.rpEntInf[i] := null
131. continue;
132. endif
133. else
134. /\* pre-LH server \*/
135. if (not (msgIn.RequiredAttrs.pAttr[j].AttrType in GCPas))
136. then
137. /\* skip requested attributes not part of GCPas \*/
138. msgOut.rpEntInf[i] := null
139. continue;
140. endif
141. endif
142. attribute := LocalAttidFromRemoteAttid(
143. msgIn.PrefixTable,
144. msgIn.RequiredAttrs.pAttr[j].attrTyp)
145. msgOut.rpEntInf[i].AttrBlock.pAttr[j].attrTyp := attribute
146. k := 0
147. foreach val in GetAttrVals(o, attribute, false)
148. msgOut.rpEntInf[i].AttrBlock.pAttr[j].AttrVal.pAVal :=
149. ADR(ATTRVALFromValue(val,
150. Syntax(attribute),
151. dc.prefixTable))
152. msgOut.rpEntInf[i].AttrBlock.pAttr[j].AttrVal.valCount :=
153. k + 1
154. endfor
155. endfor
156. else
157. msgOut.rpEntInf[i] := null
158. endif
159. endif
160. endfor /\* i := \*/
161. pmsgOut^.V1 := msgOut
162. return 0

#### Examples of the IDL\_DRSVerifyNames Method

##### Initial State

Querying the [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17) JaneDow on DC=CONTOSO, DC=COM

* ldap\_search\_s("CN=JaneDow,CN=Users,DC=contoso,DC=com", *baseObject*, "(objectClass=\*)", [*objectGUID, objectSid, sAMAccountName, sAMAccountType*])
* Getting 1 entries:
* >> Dn: CN=JaneDow,CN=Users,DC=contoso,DC=com
  + 1> objectGUID: 772cf177-00f8-45ed-9c72-5e5206bead02;
  + 1> objectSid: S-1-5-21-3263199975-614030967-162443871-1603;
  + 1> sAMAccountName: JaneDow;
  + 1> sAMAccountType: SAM\_NORMAL\_USER\_ACCOUNT;

##### Client Request

To get a user's [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d), DC2 invokes the [IDL\_DRSVerifyNames](#Section_80739a29e8ed44788490475a18e9779d) method against DC1 with the following parameters ([DRS\_HANDLE](#Section_55fed7de17f54ff38c53866df925056a) to DC1 omitted):

* *dwInVersion* = 1
* *pmsgIn* = [DRS\_MSG\_VERIFYREQ\_V1](#Section_4593a76b71f14e4cb5e14d2e27afb3cb)
  + dwFlags: 2
  + cNames: 1
  + rpNames: [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86)
    - StringName: "CN=Jane Dow,CN=Users,DC=contoso,DC=com"
  + RequiredAttrs: [ATTRBLOCK](#Section_f81324b8640041b5bc255117589c602a)
    - attrCount: 3
    - pAttr: [ATTR](#Section_a2db41e278034d3ca4990fee92b1c149)
      * sAMAccountType
      * objectSid
      * sAMAccountName

##### Server Response

The server responds with a return code of 0 and the following values:

* *pMsgOut* = DRS\_MSG\_VERIFYREPLY\_V1
  + cNames: 1
  + rpEntInf: [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b)
  + pName: [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86)
    - Guid: GUID {772cf177-00f8-45ed-9c72-5e5206bead02}
    - SID: S-1-5-21-3263199975-614030967-162443871-1603
    - String Name: "CN=Jane Dow,CN=Users,DC=contoso,DC=com"
  + ulFlags: ENTINF\_FROM\_MASTER
  + AttrBlock: [ATTRBLOCK](#Section_f81324b8640041b5bc255117589c602a)
    - sAMAccountType: 0x30000000
    - objectSid: S-1-5-21-3263199975-614030967-162443871-1603
    - sAMAccountName: JaneDow
  + PrefixTable: [SCHEMA\_PREFIX\_TABLE](#Section_9b371267e8b84c69997902dae02e5e38)

##### Final State

No change in state.

### IDL\_DRSWriteSPN (Opnum 13)

The IDL\_DRSWriteSPN method [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the set of [**SPNs**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) on an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

1. ULONG IDL\_DRSWriteSPN(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_SPNREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_SPNREPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message. Must be set to 1, because that is the only version supported.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message. The value must be 1 because that is the only version supported.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if a failure occurs.

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_SPNREQ

The DRS\_MSG\_SPNREQ union defines the request messages sent to the [IDL\_DRSWriteSPN](#Section_8b129dc8ed4545379555b6fef764ab7d) method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_SPNREQ\_V1 V1;
6. } DRS\_MSG\_SPNREQ;

**V1:**  The version 1 request.

##### DRS\_MSG\_SPNREQ\_V1

The DRS\_MSG\_SPNREQ\_V1 structure defines a request message sent to the [IDL\_DRSWriteSPN](#Section_8b129dc8ed4545379555b6fef764ab7d) method.

1. typedef struct {
2. DWORD operation;
3. DWORD flags;
4. [string] const WCHAR\* pwszAccount;
5. [range(0,10000)] DWORD cSPN;
6. [string, size\_is(cSPN)] const WCHAR\*\* rpwszSPN;
7. } DRS\_MSG\_SPNREQ\_V1;

**operation:**  The [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) operation to perform. MUST be one of the DS\_SPN\_OPERATION values.

**flags:**  Unused. MUST be 0 and ignored.

**pwszAccount:**  The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to modify.

**cSPN:**  The number of items in the **rpwszSPN** array.

**rpwszSPN:**  The SPN values.

##### DRS\_MSG\_SPNREPLY

The DRS\_MSG\_SPNREPLY union defines the response messages received from the [IDL\_DRSWriteSPN](#Section_8b129dc8ed4545379555b6fef764ab7d) method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_SPNREPLY\_V1 V1;
6. } DRS\_MSG\_SPNREPLY;

**V1:**  The version 1 response.

##### DRS\_MSG\_SPNREPLY\_V1

The DRS\_MSG\_SPNREPLY\_V1 structure defines a response message received from the [IDL\_DRSWriteSPN](#Section_8b129dc8ed4545379555b6fef764ab7d) method.

1. typedef struct {
2. DWORD retVal;
3. } DRS\_MSG\_SPNREPLY\_V1;

**retVal:**  0, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

##### DS\_SPN\_OPERATION

The DS\_SPN\_OPERATION type indicates the operation to perform.

This type is declared as follows:

1. typedef DWORD DS\_SPN\_OPERATION;

It must be one of the following values.

| Value | Meaning |
| --- | --- |
| DS\_SPN\_ADD\_SPN\_OP  (0x00000000) | Adds the specified values to the existing set of [**SPNs**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4). |
| DS\_SPN\_REPLACE\_SPN\_OP  (0x00000001) | Removes all the existing SPNs, then adds the specified values. If the set of specified values is empty (**cSPN** is zero), no values are added. |
| DS\_SPN\_DELETE\_SPN\_OP  (0x00000002) | Removes all the existing SPNs. |

#### Method-Specific Abstract Types and Procedures

##### ExecuteWriteSPNRemotely

1. procedure ExecuteWriteSPNRemotely(
2. DWORD dwInVersion,
3. DRS\_MSG\_SPNREQ \*pmsgIn,
4. DWORD \*pdwOutVersion,
5. DRS\_MSG\_SPNREPLY \*pmsgOut): ULONG

This procedure is executed only on an [**RODC**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870). It finds a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that holds a [**full NC replica**](#gt_f523a137-bda8-45a0-8c9b-f54d86b00bcb) of the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) of the RODC, performs the [IDL\_DRSWriteSPN](#Section_8b129dc8ed4545379555b6fef764ab7d) [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) method call with the given parameters against the DC in the client's [**security context**](#gt_88d49f20-6c95-4b64-a52c-c3eca2fe5709), and returns the value returned by that RPC call.

#### Server Behavior of the IDL\_DRSWriteSPN Method

*Informative summary of behavior*: The [IDL\_DRSWriteSPN](#Section_8b129dc8ed4545379555b6fef764ab7d) method [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the servicePrincipalName [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). The values of this multivalued attribute are called [**service principal names (SPNs)**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4). The IDL\_DRSWriteSPN method does one of three things:

* Adds a non-empty set of SPNs to the object's servicePrincipalName. If a member of the set is already present on the object's servicePrincipalName, it is ignored.
* Removes all current values from the object's servicePrincipalName, then adds a (possibly empty) set of SPNs to the object's servicePrincipalName.
* Removes a non-empty set of SPNs from the object's servicePrincipalName. If a member of the set is not present on the object's servicePrincipalName, it is ignored.

The effect of this method can be achieved by an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) Modify operation to the servicePrincipalName attribute of an object. Some manipulations of the servicePrincipalName attribute that cannot be performed using this method can be performed using LDAP Modify. For example, an LDAP Modify can remove one specific SPN from the servicePrincipalName attribute while adding another SPN to the servicePrincipalName attribute in the same transaction; IDL\_DRSWriteSPN cannot do this.

1. ULONG
2. IDL\_DRSWriteSPN(
3. [in, ref] DRS\_HANDLE hDrs,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DRS\_MSG\_SPNREQ \*pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DRS\_MSG\_SPNREPLY \*pmsgOut);
10. accountDN: unicodestring
11. account: DSName
12. err: DWORD
13. operation: DS\_SPN\_OPERATION
14. cSPN: integer
15. spnSet: set of unicodestring
16. instanceName: unicodestring
17. ValidateDRSInput(hDrs, 13)
18. pdwOutVersion^ := 1
19. pmsgOut^.V1.retVal := 0
20. /\* Input parameter validation \*/
21. if dwInVersion ≠ 1 then
22. pmsgOut^.V1.retVal := ERROR\_INVALID\_PARAMETER
23. return ERROR\_INVALID\_PARAMETER
24. endif
25. /\* Input parameter validation \*/
26. if ClientUUID(hDrs) ≠ NTDSAPI\_CLIENT\_GUID
27. pmsgOut^.V1.retVal := ERROR\_INVALID\_PARAMETER
28. return ERROR\_INVALID\_PARAMETER
29. endif
30. /\* RODCs do not perform originating updates \*/
31. if AmIRODC() then
32. return ExecuteWriteSPNRemotely(dwInVersion,
33. pmsgIn, pdwOutVersion, pmsgOut);
34. endif
35. accountDN := pmsgIn^.V1.pwszAccount
36. operation := pmsgIn^.V1.operation
37. cSPN := pmsgIn^.V1.cSPN
38. spnSet := pmsgIn^.V1.rpwszSPN
39. if accountDN = null or accountDN = "" then
40. pmsgOut^.V1.retVal := ERROR\_INVALID\_PARAMETER
41. return ERROR\_INVALID\_PARAMETER
42. endif
43. if not operation in [DS\_SPN\_ADD\_SPN\_OP .. DS\_SPN\_DELETE\_SPN\_OP] then
44. pmsgOut^.V1.retVal := ERROR\_INVALID\_FUNCTION
45. return ERROR\_INVALID\_FUNCTION
46. endif
47. /\* DS\_SPN\_REPLACE\_SPN\_OP permits 0 SPNs to be specified (meaning
48. \* "delete all SPNs"). Other operations require >=1 SPNs to be
49. \* specified. \*/
50. if (operation ≠ DS\_SPN\_REPLACE\_SPN\_OP) and (cSPN = 0) then
51. pmsgOut^.V1.retVal := ERROR\_INVALID\_PARAMETER
52. return ERROR\_INVALID\_PARAMETER
53. endif
54. /\* The empty string is an invalid SPN. \*/
55. foreach spn in spnSet
56. if spn = null or spn = "" then
57. pmsgOut^.V1.retVal := ERROR\_INVALID\_PARAMETER
58. return ERROR\_INVALID\_PARAMETER
59. endif
60. endfor
61. account := GetDSNameFromDN(accountDN);
62. if not ObjExists(account) then
63. pmsgOut^.V1.retVal := ERROR\_DS\_OBJ\_NOT\_FOUND
64. return ERROR\_DS\_OBJ\_NOT\_FOUND
65. endif
66. /\* Perform access checks \*/
67. err = AccessCheckWriteToSpnAttribute(account, spnSet)
68. if err ≠ ERROR\_SUCCESS then
69. pmsgOut^.V1.retVal := err
70. return err
71. endif
72. if (operation = DS\_SPN\_DELETE\_SPN\_OP) then
73. /\* Remove specified SPNs \*/
74. foreach spn in spnSet
75. if spn in account!servicePrincipalName then
76. account!servicePrincipalName :=
77. account!servicePrincipalName - {spn}
78. endif
79. endfor
80. return 0
81. endif
82. if (operation = DS\_SPN\_ADD\_SPN\_OP) then
83. /\* Add specified SPNs \*/
84. foreach spn in spnSet
85. account!servicePrincipalName :=
86. account!servicePrincipalName + {spn}
87. endfor
88. return 0
89. endif
90. /\* Must be DS\_SPN\_REPLACE\_SPN\_OP.
91. \* Remove all existing SPNs, then add in the specified SPNs. \*/
92. account!servicePrincipalName := {null}
93. foreach spn in spnSet
94. account!servicePrincipalName :=
95. account!servicePrincipalName + {spn}
96. endfor
97. return 0

### IDL\_DRSAddCloneDC (Opnum 28)

The IDL\_DRSAddCloneDC method is used to create a new [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) by copying [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) from an existing DC object.

1. ULONG IDL\_DRSAddCloneDC(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_ ADDCLONEDCREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_ ADDCLONEDCREPLY\* pmsgOut
9. );

**hDrs:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, otherwise a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_ADDCLONEDCREQ

The DRS\_MSG\_ADDCLONEDCREQ union defines the request messages sent to the [IDL\_DRSAddCloneDC](#Section_ef0bfb1d037b4626a6d9cc7589bc5786) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_ADDCLONEDCREQ\_V1 V1;
6. } DRS\_MSG\_ADDCLONEDCREQ;

**V1:**  The version 1 request.

##### DRS\_MSG\_ADDCLONEDCREQ\_V1

The DRS\_MSG\_ADDCLONEDCREQ\_V1 structure defines a request message sent to the [IDL\_DRSAddCloneDC](#Section_ef0bfb1d037b4626a6d9cc7589bc5786) method.

1. typedef struct {
2. [string] const WCHAR\* pwszCloneDCName;
3. [string] const WCHAR\* pwszSite;
4. } DRS\_MSG\_ADDCLONEDCREQ\_V1;

**pwszCloneDCName:**  The new [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) name.

**pwszSite:**  The [**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) of the [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba) the new DC will be placed into.

##### DRS\_MSG\_ADDCLONEDCREPLY

The DRS\_MSG\_ADDCLONEDCREPLY union defines the response messages received from the [IDL\_DRSAddCloneDC](#Section_ef0bfb1d037b4626a6d9cc7589bc5786) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_ADDCLONEDCREPLY\_V1 V1;
6. } DRS\_MSG\_ADDCLONEDCREPLY;

**V1:**  The version 1 response.

##### DRS\_MSG\_ADDCLONEDCREPLY\_V1

The DRS\_MSG\_ADDCLONEDCREPLY\_V1 structure defines a response message received from the [IDL\_DRSAddCloneDC](#Section_ef0bfb1d037b4626a6d9cc7589bc5786) method.

1. typedef struct {
2. [string] WCHAR\* pwszCloneDCName;
3. [string] WCHAR\* pwszSite;
4. [range(0,1024)] DWORD cPasswordLength;
5. [size\_is(cPasswordLength)] WCHAR\* pwsNewDCAccountPassword;
6. } DRS\_MSG\_ADDCLONEDCREPLY\_V1;

**pwszCloneDCName:**  The new [**DC's**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) name.

**pwszSite:**  The [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba) containing the new DC.

**cPasswordLength:**  The length of the **pwsNewDCAccountPassword** member.

**pwsNewDCAccountPassword:**  The password of the new DC account.

#### Method-Specific Abstract Types and Procedures

##### GetKeyLength

1. procedure GetKeyLength(hDrs: DRS\_HANDLE): integer

Returns the key length, in bits, of the encryption used on the *hDrs* connection. Returns 0 if no encryption is in use on the connection.

##### DNMap

1. type DNMap : Map {originalObj : DSName} to {newObj : DSName}

A map from one [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) to another DN.

##### DCInfo

This [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) stores information about a [**domain controller**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

1. typedef struct {
2. string Name;
3. string dnsHostName;
4. SID Sid;
5. } DCInfo;

**Name**: The DC's name.

**dnsHostName**: The DC's DNS host name.

**Sid**: The DC's [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d).

##### TranslationInfo

Represents translation from the original [**domain controller**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) to the new domain controller.

1. typedef struct {
2. DCInfo OriginalDC;
3. DCInfo NewDC;
4. DNMap objMap;
5. } TranslationInfo;

**originalDC**: The original DC's information.

**newDC**: The new DC's information.

**objMap**: The map of the original DC-related [**DNs**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) to the new DC-related DN.

##### ReplaceName

1. ReplaceName(stringValue : string, originalName : string, newName : string) : string

Replaces all occurrences of *originalName* in *stringValue* with *newName*, and returns the resulting string.

##### ReplaceSIDInSecurityDescriptor

1. ReplaceSIDInSecurityDescriptor(sd : SECURITY\_DESCRIPTOR, originalSid : SID, newSid : SID)
2. : SECURITY\_DESCRIPTOR

Creates a copy of [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) *sd*, replaces all occurrences of *originalSid* with *newSid* in the security descriptor, and returns the new security descriptor.

##### GetPrincipalSid

1. GetPrincipalSid(clientCreds : ClientAuthorizationInfo) : SID

Returns the user-[**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) part of the *clientCreds* [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b).

##### GenerateNewKrbTgtAcct

1. GenerateNewKrbTgtAcct() : DSName

Generates a Kerb Tgt user account in the [**local domain controller (local DC)**](#gt_17b69a5a-adc1-4763-92cf-5e44f11abbe7) using the same steps as [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.3.4.1.23. The following steps are performed by this abstract procedure:

* Creates a new [**user object**](#gt_e767a471-c3fa-4e4b-a40c-daeb08f82a17).
* Selects a value in the range [1 .. 65535] that is not currently present as a value of the msDS-SecondaryKrbTgtNumber [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on any [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in this [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), and assigns the value to the msDS-SecondaryKrbTgtNumber attribute of the created object. If no such value exists, the result is the error *other* / *ERROR\_NO\_SYSTEM\_RESOURCES*.
* The selected value for msDS-SecondaryKrbTgtNumber is appended (in decimal form) to the string "krbtgt", and the resulting string is assigned to the sAMAccountName attribute on the created object.
* The userAccountControl bits ADS\_UF\_ACCOUNT\_DISABLE and ADS\_UF\_DONT\_EXPIRE\_PASSWD are set on the object's userAccountControl attribute.
* The object's account password is set to a randomly generated value that satisfies all criteria in [[MS-SAMR]](%5bMS-SAMR%5d.pdf#Section_4df07fab1bbc452f8e927853a3c7e380) section 3.1.1.7.2 and is processed as described in [MS-SAMR] section 3.1.1.8.5.
* Returns the [**DSName**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a) of the created object.

##### DuplicateObject

1. Procedure DuplicateObject (
2. originalObj : DSName,
3. newObjParent : DSName,
4. newObjRdn : string,
5. tlInfo : TranslationInfo) : DSName

*Informative summary of behavior*: This procedure creates a new [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) by copying data from an existing object. When copying data, it replaces any reference to the original [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) in the object data with a reference to the new DC. The new object is created under *newObjectParent* and its [**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) is set to *newObjRdn*.

1. Procedure DuplicateObject (
2. originalObj : DSName,
3. newObjParent : DSName,
4. newObjRdn : string,
5. tlInfo : TranslationInfo) : DSName
6. newObj : DSName
7. forwardLinkAttribute : string
8. referenceObj : DSName
9. newObj!distinguishedName := newObjRdn + ',' +
10. newObjParent!distinguishedName
11. foreach attribute in originalObj!attr
12. if attribute in {
13. objectClass, objectCategory, userAccountControl,
14. hasMasterNCs, msDS-hasMasterNCs, dMDLocation, msDS-HasDomainNCs,
15. options, systemFlags, showInAdvancedViewOnly,
16. msDS-NeverRevealGroup, msDS-RevealOnDemandGroup,
17. msDS-RevealedUsers, managedBy, msDS-Behavior-Version,
18. msDS-HasDomainNCs, msDS-hasFullReplicaNCs, enabledConnection,
19. fromServer} then
20. newObj!attribute := originalObj!attribute
21. else if attribute in {sAMAccountName, dNSHostName} then
22. newObj!attribute.Value := ReplaceName(originalObj!attribute.Value,
23. tlInfo.originalDC.Name, tlInfo.newDC.Name)
24. else if attribute in {serverReference, msDS-KrbTgtLink,
25. msDFSR-ComputerReference} then
26. /\* replace reference to original DC-related object with new
27. DC object using objMap\*/
28. newObj!attribute.Value :=
29. tlInfo.objMap[originalObj!distinguishedName]
30. else if attribute = servicePrincipalName then
31. foreach servicePrincipalName in originalDC!servicePrincipalName
32. newServicePrincipalName : string
33. newServicePrincipalName := servicePrincipalName
34. if newServicePrincipalName contains tlInfo.OriginalDC.Name then
35. newServicePrincipalName :=
36. ReplaceName(newServicePrincipalName,
37. tlInfo.OriginalDC.Name,
38. tlInfo.NewDC.Name)
39. newObj!servicePricipalName :=
40. newObj!servicePricipalName +
41. {newServicePrincipalName}
42. else if newServicePrincipalName
43. contains(tlInfo.originalDC.dnsHostName) then
44. newServicePrincipalName :=
45. ReplaceName(newServicePrincipalName,
46. tlInfo.OriginalDC.dnsHostName,
47. tlInfo.newDC.dnsHostName)
48. newObj!servicePricipalName :=
49. newObj!servicePricipalName +
50. {newServicePrincipalName}
51. endif
52. endfor
53. else if attribute = invocationId then
54. newObj!invocationId := a random guid
55. else if attribute = nTSecurityDescriptor then
56. if tlInfo.newDC.Sid ≠ null then
57. newObj!nTSecurityDescriptor := ReplaceSIDInSecurityDescriptor (
58. originalDC!nTSecurityDescriptor, tlInfo.originalDC.Sid,
59. tlInfo.newDC.Sid)
60. endif
61. endif
62. endfor
63. /\* If a back link points to the original DC object, update the forward
64. link in the referenced object
65. \*/
66. foreach attribute in originalObj!Attributes
67. if attribute in {memberOf, msDS-NC-RO-Replica-Locations-BL} then
68. if attribute = isMemberOf then
69. forwardLinkAttribute := member
70. else if attribute = msDS-NC-RO-Replica-Locations-BL then
71. forwardLinkAttribute := msDS-NC-RO-Replica-Locations
72. endIf
73. if tlInfo.objMap.Keys.exists(originalObj!attribute) then
74. referenceObj := tlInfo.objMap[originalObj!attribute]
75. else
76. referenceObj := select o from all
77. where o!distinguishedName = originalObj!attribute
78. endif
79. referenceObj!forwardLinkAttribute := newObj
80. endif
81. endfor
82. return newObj

#### Server Behavior of the IDL\_DRSAddCloneDC Method

*Informative summary of behavior*: The [IDL\_DRSAddCloneDC](#Section_ef0bfb1d037b4626a6d9cc7589bc5786) method is used to create a new [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) by duplicating the states of the original DC. The states of a DC are composed of [**computer**](#gt_d8e8f5a7-db85-40a8-98ed-1abab2237b82), server, NTDS settings, FRS, DFSR, and connection [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that are maintained for each DC. When duplicating an object, this [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) method replaces all references to the original DC with corresponding objects of the new DC. The caller has to have the [**control access right**](#gt_42f6c9e0-a2b3-4bc3-9b87-fdb902e5505e) DS-Clone-Domain-Controller on the [**default NC**](#gt_adc2c434-9679-419f-8c8b-b8c234921ad3). When called, this RPC method:

1. Validates that the caller has permission to perform the operation.
2. Creates new account and other objects for the new domain controller account by copying information from the existing domain controller.
3. Returns the name, [**site**](#gt_8abdc986-5679-42d9-ad76-b11eb5a0daba), and password for the new domain controller to the client.
4. ULONG
5. IDL\_DRSAddCloneDC(
6. [in, ref] DRS\_HANDLE hDrs,
7. [in] DWORD dwInVersion,
8. [in, ref, switch\_is(dwInVersion)]
9. DRS\_MSG\_ADDCLONEDCREQ \*pmsgIn,
10. [out, ref] DWORD \*pdwOutVersion,
11. [out, ref, switch\_is(\*pdwOutVersion)]
12. DRS\_MSG\_ADDCLONEDCREPLY \*pmsgOut)
13. msgIn: DRS\_MSG\_ADDCLONEDCREQ\_V1
14. clientCreds: ClientAuthorizationInfo
15. tlInfo: TranslationInfo
16. callerSid: SID
17. isRodc: boolean
18. computerObj: DSName
19. originalDCSrvObj: DSName
20. originalDCSiteObj: DSName
21. originalDCServersObj: DSName
22. originalDSAObj: DSName
23. newDCComputerObj: DSName
24. newDCSiteObj: DSName
25. newDCServersObj: DSName
26. newDCServerObj: DSName
27. newDSAObj: DSName
28. ValidateDRSInput(hDrs, 28)
29. pdwOutVersion^ := 1
30. pmsgOut^.V1.pwszCloneDCName := null
31. pmsgOut^.V1.pwszSite := null
32. pmsgOut^.V1.cPasswordLength := 0
33. pmsgOut^.V1.pwsNewDCAccountPassword := null
34. if dwInVersion ≠ 1 then
35. return ERROR\_DS\_DRA\_INVALID\_PARAMETER
36. endif
37. msgIn := pmsgIn^.V1
38. if GetKeyLength(hDrs) < 128 then
39. return ERROR\_DS\_STRONG\_AUTH\_REQUIRED
40. endif
41. if not AccessCheckCAR(DefaultNC(), DS-Clone-Domain-Controller) then
42. return ERROR\_DS\_DRA\_ACCESS\_DENIED
43. endif
44. /\* Check that the caller (the "source" DC) is actually a DC by
45. \* checking Enterprise Domain Controllers or Enterprise Read-Only Domain
46. \* Controllers SID in its token. \*/
47. clientCreds := GetCallerAuthorizationInfo()
48. if not CheckGroupMembership(clientCreds, SidFromStringSid("S-1-5-9")) then
49. if not CheckGroupMembership(clientCreds, SidFromStringSid("S-1-5-498"))
50. then
51. return ERROR\_DS\_DRA\_ACCESS\_DENIED
52. else
53. isRodc := true
54. endif
55. endif
56. /\* The DC must own the PDC role \*/
57. if GetFSMORoleOwner(FSMO\_PDC) ≠ DSAObj() then
58. return ERROR\_INVALID\_DOMAIN\_ROLE
59. endif
60. callerSid := GetPrincipalSid(clientCreds)
61. /\* get the original DC computer object \*/
62. computerObj := select one obj from all where
63. (obj!objectSid = callerSid)
64. tlInfo.OriginalDC.Name := computerObj!sAMAccountName.Remove('$')
65. /\* generate cloned DC name if not specified \*/
66. if (msgIn.pwszCloneDCName = null)
67. found : boolean
68. newDCName : string
69. /\* Generate new name by appendling '–CL' and 4 digits to the original
70. \* DC name \*/
71. found := false
72. For suffix = 0000 to 9999 do
73. newDCName := tlInfo.OriginalDC.Name[0 .. 8] + '-CL' + suffix
74. if not exists(
75. select o from all where o!sAMAccountName = (newDCName + '$')
76. ) then
77. found := true
78. break
79. endif
80. endfor
81. if not found then
82. return ERROR\_DS\_UNWILLING\_TO\_PERFORM
83. endif
84. tlInfo.newDC.Name := newDCName
85. else
86. tlInfo.newDC.Name := msgIn.pwszCloneDCName
87. endIf
88. tlInfo.OriginalDC.Sid := computerObj!objectSid
89. tlInfo.OriginalDC.dnsHostName := computerObj!dNSHostName
90. if isRodc then
91. newKrbTgtAcct : DSName
92. newKrbTgtAcct := GenerateNewKrbTgtAcct()
93. tlInfo.objMap[computerObj!msDS-KrbTgtLink] := newKrbTgtAcct
94. endif
95. /\* Duplicate original DC computer object \*/
96. newDCComputerObj := DuplicateObject(computerObj, computerObj!parent,
97. "cn=" + tlInfo.newDC.Name, tlInfo)
98. tlInfo.objMap[compuerObj!distinguishedName] :=
99. newDCComputerObj!distinguishedName
100. tlInfo.NewDC.Sid := newDCComputerObj!objectSid
101. tlInfo.NewDC.dnsHostName := newDCComputerObj!dNSHostName
102. /\* Get the original DC server object \*/
103. originalDCSrvObj := select one v from ConfigNC()
104. where v.dn in computerObj!serverReferenceBL
105. originalDCServersObj := originalDCSrvObj!parent
106. originalDCSiteObj := originalDCSrvObj!parent
107. /\* use the specified site for the new DC.
108. \* use the original DC site if the site is not specified \*/
109. if (msgIn.pwszSite ≠ null) then
110. siteContainer: DSName
111. siteContainer := DescendantObject(ConfigNC(), "CN=Sites,")
112. newDCSiteObj := select one v from siteContainer!children
113. where v!name = msgIn.pwszSite
114. if newDCSiteObj = null
115. return ERROR\_NO\_SUCH\_SITE
116. endIf
117. else
118. newDCSiteObj := originalDCSiteObj
119. endIf
120. newDCServersObj := DescendantObject(newDCSiteObj, "CN=Servers")
121. /\* Duplicate the original DC servers object if the servers object is not
122. \* present in the new DC site \*/
123. if not exists newDCServersObj then
124. newDCServersObj := DuplicateObject(originalDCServersObj,
125. newDCSiteObj, "CN=Servers", tlInfo)
126. endIf
127. tlInfo.objMap[originalDCServersObj!distinguishedName] :=
128. newDCServersObj!distinguishedName
129. /\* Duplicate the server object \*/
130. newDCServerObj := DescendantObject(newDCServersObj,
131. "CN=" + tlInfo.newDC.Name)
132. if not exists newDCServerObj then
133. newDCServerObj := DuplicateObject(originalDCSrvObj, newDCServersObj,
134. "CN=" + tlInfo.newDC.Name, tlInfo)
135. endIf
136. tlInfo.objMap[originalDCSrvObj!distinguishedName] :=
137. newDCServerObj!distinguishedName
138. /\* Duplicate the NTDS settings object \*/
139. originalDSAObj := DescendantObject(originalDCSrvObj,
140. "CN=NTDS Settings")
141. newDSAObj := DuplicateObject(originalDSAObj, newDCServerObj,
142. "CN=NTDS Settings", tlInfo)
143. tlInfo.objMap[originalDSAObj!distinguishedName] :=
144. newDSAObj!distinguishedName
145. if isRodc then
146. newConnObj: DSName
147. topologyObj: DSName
148. originalDFSRObj: DSName
149. newDFSRObj: DSName
150. frsSysvolObj: DSName
151. originalFRSObj: DSName
152. newfrsObj : DSName
153. foreach obj in originalDSAObj!children where
154. obj!objectClass = "ntdsConnection"
155. newConnObj := DuplicateObject(obj, newDSAObj,
156. "CN=" + tlInfo.newDC.Name, tlInfo)
157. objMap[obj!distinguishedName] := newConnObj!distinguishedName
158. endfor
159. /\* Duplicate DFSR topology object \*/
160. topologyObj := DescendantObject(DefaultNC(),
161. "CN=Topology,CN=Domain System Volume,CN=DFSR-GlobalSettings,CN=System")
162. originalDFSRObj := DescendantObject(topologyObj,
163. "CN=" + tlInfo.OriginalDC.Name)
164. if originalDFSRObj ≠ null then
165. newDFSRObj = DuplicateObject(originalDFSRObj, topologyObj,
166. "CN="+ tlInfo.newDC.Name, tlInfo)
167. endIf
168. /\* Duplicate FRS object \*/
169. frsSysvolObj = DescendantObject(DefaultNC(),
170. "CN=Domain System Volume (SYSVOL share),CN=File Replication Service,CN=System")
171. originalFRSObj = DescendantObject(frsSysvolObj,
172. "CN=" + tlInfo.OriginalDC.Name)
173. if originalFRSObj ≠ null then
174. newfrsObj = DuplicateObject(originalFRSObj, frsSysVolObj,
175. "CN=" + tlInfo.newDC.Name, tlInfo)
176. endIf
177. endif
178. pmsgOut^.V1.pwszCloneDCname := tlInfo.newDC.Name
179. pmsgOut^.V1.cPasswordLength := 120
180. pmsgOut^.V1.pwsNewDCAccountPassword:= a 120-byte sequence of randomly
181. generated characters between ASCII 32 (space) and ASCII 122 ('z')
182. pmsgOut^.V1.pwszSite := newDCSiteObj!name
183. return 0

#### Examples of the IDL\_DRSAddCloneDC Method

DC1 invokes the [IDL\_DRSAddCloneDC](#Section_ef0bfb1d037b4626a6d9cc7589bc5786) method on the [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) to create a cloned [**domain controller**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) account "DC1Clone1" in the CONTOSO.COM [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef).

##### Initial State

Querying the DC1 [**computer object**](#gt_d8e8f5a7-db85-40a8-98ed-1abab2237b82) in [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) DC=CONTOSO, DC=COM by performing an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) search with base scope on the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) "CN=DC1,OU=Domain Controllers,DC=contoso,DC=com":

* Expanding base 'CN=DC1,OU=Domain Controllers,DC=contoso,DC=com'...
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=DC1,OU=Domain Controllers,DC=contoso,DC=com
  + 5> objectClass: top; person; organizationalPerson; user; computer;
  + 1> cn: DC1;
  + 1> distinguishedName: CN=DC1, OU=Domain Controllers, DC=contoso, DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/10/2006 18:04:35 Pacific Standard Daylight Time;
  + 1> whenChanged: 07/15/2006 19:39:05 Pacific Standard Daylight Time;
  + 1> uSNCreated: 12291;
  + 1> uSNChanged: 24577;
  + 1> name: DC1;
  + 1> objectGUID: ac1993e1-0377-4161-893e-ccd2a98e1bba;
  + 1> userAccountControl: (UF\_SERVER\_TRUST\_ACCOUNT | UF\_TRUSTED\_FOR\_DELEGATION );
  + 1> badPwdCount: 0;
  + 1> codePage: 0;
  + 1> countryCode: 0;
  + 1> badPasswordTime: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogoff: 01/01/1601 00:00:00 UNC ;
  + 1> lastLogon: 07/17/2006 19:47:40 Pacific Standard Daylight Time;
  + 1> localPolicyFlags: 0;
  + 1> pwdLastSet: 07/10/2006 18:04:35 Pacific Standard Daylight Time;
  + 1> primaryGroupID: 516;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1001;
  + 1> accountExpires: 09/14/30828 02:48:05 UNC ;
  + 1> logonCount: 17;
  + 1> sAMAccountName: DC1$;
  + 1> sAMAccountType: 805306369;
  + 1> operatingSystem: Windows Server 2003;
  + 1> operatingSystemVersion: 5.2 (3790);
  + 1> operatingSystemServicePack: Service Pack 1;
  + 1> serverReferenceBL: CN=DC1,CN=Servers, CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dNSHostName: DC1.contoso.com;
  + 1> rIDSetReferences: CN=RID Set,CN=DC1,OU=Domain Controllers, DC=contoso, DC=com;
  + 15> servicePrincipalName: ldap/DC1.contoso.com/NDNC5.contoso.com; ldap/DC1.contoso.com/NDNC2.contoso.com; ldap/DC1.contoso.com/NDNC1.contoso.com; GC/DC1.contoso.com/contoso.com; HOST/DC1.contoso.com/CONTOSO; HOST/DC1; HOST/DC1.contoso.com; HOST/DC1.contoso.com/contoso.com; E3514235-4B06-11D1-AB04-00C04FC2DCD2/c20bc312-4d35-4cc0-9903-b1073368af4a/contoso.com; ldap/c20bc312-4d35-4cc0-9903-b1073368af4a.\_msdcs.contoso.com; ldap/DC1.contoso.com/CONTOSO; ldap/DC1; ldap/DC1.contoso.com; ldap/DC1.contoso.com/contoso.com; NtFrs-88f5d2bd-b646-11d2-a6d3-00c04fc9b232/DC1.contoso.com;
  + 1> objectCategory: CN=Computer, CN=Schema, CN=Configuration, DC=contoso, DC=com;
  + 1> isCriticalSystemObject: TRUE;
  + 1> frsComputerReferenceBL: CN=DC1, CN=Domain System Volume (SYSVOL share),CN=File Replication Service,CN=System,DC=contoso,DC=com;
  + 1> lastLogonTimestamp: 07/11/2006 04:02:42 Pacific Std Daylight Time;

Querying the DC1 configuration [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625) CN=Configuration, DC=CONTOSO, DC=COM by performing an LDAP search with subtree scope on the DN "CN=DC1, CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com":

* ldap\_search\_s(ld, "CN=ALPHA10,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com", 2, "(objectClass=\*)", attrList, 0, &msg)
* Getting 5 entries:
* >>Dn: CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + 1> cn: DC1;
  + 1> distinguishedName: CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dNSHostName: DC1.mohkhan-TEST10.nttest.microsoft.com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = ( WRITE );
  + 1> name: DC1;
  + 1> objectCategory: CN=Server,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; server;
  + 1> objectGUID: 75568225-7ec6-4d83-a72d-82d19c0799c5;
  + 1> serverReference: CN=DC1,OU=Domain Controllers,DC=contoso,DC=com;
  + 1> showInAdvancedViewOnly: TRUE;
  + 1> systemFlags: 0x52000000 = ( CONFIG\_ALLOW\_RENAME | CONFIG\_ALLOW\_LIMITED\_MOVE | DISALLOW\_MOVE\_ON\_DELETE );
  + 1> uSNChanged: 7763;
  + 1> uSNCreated: 7747;
  + 1> whenChanged: 7/21/2011 2:51:29 PM Pacific Daylight Time;
  + 1> whenCreated: 7/21/2011 2:18:57 PM Pacific Daylight Time;
* >> Dn: CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + 1> cn: NTDS Settings;
  + 1> distinguishedName: CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dMDLocation: CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 3> hasMasterNCs: CN=Schema,CN=Configuration,DC=contoso,DC=com; CN=Configuration,DC=contoso,DC=com; DC=contoso,DC=com;
  + 1> instanceType: 0x4 = ( WRITE );
  + 1> invocationId: 64ef4da2-e442-4d8b-98d9-933609051bec;
  + 1> msDS-Behavior-Version: 5;
  + 1> msDS-HasDomainNCs: DC=contoso,DC=com;
  + 3> msDS-HasInstantiatedNCs: B:8:0000000D:CN=Schema,CN=Configuration,DC=contoso,DC=com; B:8:0000000D:CN=Configuration,DC=contoso,DC=com; B:8:00000005:DC=contoso,DC=com;
  + 3> msDS-hasMasterNCs: N=Schema,CN=Configuration,DC=contoso,DC=com; CN=Configuration,DC=contoso,DC=com; DC=contoso,DC=com;
  + 1> name: NTDS Settings;
  + 1> objectCategory: CN=NTDS-DSA,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 3> objectClass: top; applicationSettings; nTDSDSA;
  + 1> objectGUID: 64ef4da2-e442-4d8b-98d9-933609051bec;
  + 1> options: 0x1 = ( IS\_GC );
  + 1> serverReferenceBL: CN=DC1,CN=Domain System Volume (SYSVOL share),CN=File Replication Service,CN=System,DC=contoso,DC=com;
  + 1> showInAdvancedViewOnly: TRUE;
  + 1> systemFlags: 0x2000000 = ( DISALLOW\_MOVE\_ON\_DELETE );
  + 1> uSNChanged: 7777;
  + 1> uSNCreated: 7755;
  + 1> whenChanged: 7/21/2011 2:51:29 PM Pacific Daylight Time;
  + 1> whenCreated: 7/21/2011 2:18:57 PM Pacific Daylight Time;
* >> Dn: CN=a00d8e30-9afb-47de-80e3-06f53ffd88bd,CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + 1> cn: a00d8e30-9afb-47de-80e3-06f53ffd88bd;
  + 1> distinguishedName: CN=a00d8e30-9afb-47de-80e3-06f53ffd88bd,CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> enabledConnection: TRUE;
  + 1> fromServer: CN=NTDS Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> instanceType: 0x4 = ( WRITE );
  + 3> mS-DS-ReplicatesNCReason: B:8:00000008:CN=Schema,CN=Configuration,DC=contoso,DC=com; B:8:00000008:DC=contoso,DC=com; B:8:00000008:CN=Configuration,DC=contoso,DC=com;
  + 3> name: a00d8e30-9afb-47de-80e3-06f53ffd88bd;
  + 1> objectCategory: CN=NTDS-Connection,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 3> objectClass: top; leaf; nTDSConnection;
  + 1> objectGUID: 9a3269e9-8bae-4bc2-a201-865d1b4785c6;
  + 1> options: 0x1 = ( IS\_GENERATED );
  + 1> schedule: Size: 188, Bandwidth: 0, NumberOfSchedules: 1, Schedules[0].Type: 0, Schedules[0].Offset: 20 1000.1000.1000.1000.... 0.1000.1000.1000.1000.1000.1000.1000.1000.1000. ;
  + 1> showInAdvancedViewOnly: TRUE;
  + 1> systemFlags: 0x60000000 = ( CONFIG\_ALLOW\_RENAME | CONFIG\_ALLOW\_MOVE );
  + 1> uSNChanged: 41448;
  + 1> uSNCreated: 12337;
  + 1> whenChanged: 7/22/2011 3:25:14 PM Pacific Daylight Time;
  + 1> whenCreated: 7/21/2011 2:54:55 PM Pacific Daylight Time;

##### Client Request

DC1 invokes the [IDL\_DRSAddCloneDC](#Section_ef0bfb1d037b4626a6d9cc7589bc5786) method against the [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) with the following parameters ([DRS\_HANDLE](#Section_55FED7DE17F54FF38C53866DF925056A) is omitted):

* *dwInVersion* = 1
* *pmsgIn* = DRS\_MSG\_ADDCLONEDCREQ\_V1
  + pwszCloneDCName = "DC1Clone1"
  + pwszSite = Site1
  + Here Site1 is an existing site.

##### Server Response

Return code of 0 and the following values:

* *pdwOutVersion^* = 1
* *pmsgOut* = DRS\_MSG\_ADDCLONEDCREPLY\_V1
  + pwszCloneDCname: DC1Clone1
  + pwszSite: Site1
  + cPasswordLength: 120
  + pwsNewDCAccountPassword: "a;adoba01>...4ei1283-0"

##### Final State

After the clone operation, [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for a new [**domain controller**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) "DC1Clone1" are present in the CONTOSO.COM [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), querying the [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) as follows:

* Expanding base 'CN=DC1CLONE1,OU=Domain Controllers,DC=contoso,DC=com'...
* Result <0>: (null)
* Matched DNs:
* Getting 1 entries:
* >> Dn: CN=DC1CLONE1,OU=Domain Controllers,DC=contoso,DC=com
  + 5> objectClass: top; person; organizationalPerson; user; computer;
  + 1> cn: DC1CLONE1;
  + 1> distinguishedName: CN=DC1CLONE1, OU=Domain Controllers, DC=contoso, DC=com;
  + 1> instanceType: 0x4 = ( IT\_WRITE );
  + 1> whenCreated: 07/27/2011 18:04:35 Pacific Standard Daylight Time;
  + 1> whenChanged: 07/27/2011 19:39:05 Pacific Standard Daylight Time;
  + 1> uSNCreated: 12291;
  + 1> uSNChanged: 24577;
  + 1> name: DC1CLONE1;
  + 1> objectGUID: b6734e82-727f-49e9-a1f5-f08ad23cb3ff;
  + 1> userAccountControl: (UF\_SERVER\_TRUST\_ACCOUNT | UF\_TRUSTED\_FOR\_DELEGATION );
  + 1> codePage: 0;
  + 1> countryCode: 0;
  + 1> localPolicyFlags: 0;
  + 1> pwdLastSet: 07/10/2006 18:04:35 Pacific Standard Daylight Time;
  + 1> primaryGroupID: 516;
  + 1> objectSid: S-1-5-21-254470460-2440132622-709970653-1051;
  + 1> accountExpires: 09/14/30828 02:48:05 UNC ;
  + 1> sAMAccountName: DC1CLONE1$;
  + 1> sAMAccountType: 805306369;
  + 1> serverReferenceBL: CN=DC1CLONE1,CN=Servers, CN=Site1,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dNSHostName: DC1CLONE1.contoso.com;
  + 15> servicePrincipalName: ldap/DC1CLONE1.contoso.com/NDNC5.contoso.com; ldap/DC1CLONE1.contoso.com/NDNC2.contoso.com; ldap/DC1CLONE1.contoso.com/NDNC1.contoso.com; GC/DC1CLONE1.contoso.com/contoso.com; HOST/DC1CLONE1.contoso.com/CONTOSO; HOST/DC1CLONE1; HOST/DC1CLONE1.contoso.com; HOST/DC1CLONE1.contoso.com/contoso.com; E3514235-4B06-11D1-AB04-00C04FC2DCD2/c20bc312-4d35-4cc0-9903-b1073368af4a/contoso.com; ldap/c20bc312-4d35-4cc0-9903-b1073368af4a.\_msdcs.contoso.com; ldap/DC1CLONE1.contoso.com/CONTOSO; ldap/DC1CLONE1; ldap/DC1CLONE1.contoso.com; ldap/DC1CLONE1.contoso.com/contoso.com; NtFrs-88f5d2bd-b646-11d2-a6d3-00c04fc9b232/DC1CLONE1.contoso.com;
  + 1> objectCategory: CN=Computer, CN=Schema, CN=Configuration, DC=contoso, DC=com;
  + 1> isCriticalSystemObject: TRUE;
  + 1> frsComputerReferenceBL: CN=DC1CLONE1, CN=Domain System Volume (SYSVOL share),CN=File Replication Service,CN=System,DC=contoso,DC=com;

Querying the DC1CLONE1 configuration objects in [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625) CN=Configuration, DC=CONTOSO, DC=COM by performing an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) search with subtree scope on the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) "CN=DC1CLONE1, CN=DC1CLONE1,CN=Servers,CN=Site1,CN=Sites,CN=Configuration,DC=contoso,DC=com":

* ldap\_search\_s(ld, "CN=ALPHA10,CN=Servers,CN=Site1,CN=Sites,CN=Configuration,DC=contoso,DC=com", 2, "(objectClass=\*)", attrList, 0, &msg)
* Getting 5 entries:
* >>Dn: CN=DC1CLONE1,CN=Servers,CN=Site1,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + 1> cn: DC1CLONE1;
  + 1> distinguishedName: CN=DC1CLONE1,CN=Servers,CN=Site1,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1> dNSHostName: DC1CLONE1.mohkhan-TEST10.nttest.microsoft.com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 1> instanceType: 0x4 = ( WRITE );
  + 1> name: DC1CLONE1;
  + 1> objectCategory: CN=Server,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 2> objectClass: top; server;
  + 1> objectGUID: 9d308291-d1c8-41cd-a32b-8aa62746a9b9;
  + 1> serverReference: CN=DC1CLONE1,OU=Domain Controllers,DC=contoso,DC=com;
  + 1> showInAdvancedViewOnly: TRUE;
  + 1> systemFlags: 0x52000000 = ( CONFIG\_ALLOW\_RENAME | CONFIG\_ALLOW\_LIMITED\_MOVE | DISALLOW\_MOVE\_ON\_DELETE );
  + 1> uSNChanged: 17763;
  + 1> uSNCreated: 17747;
  + 1> whenChanged: 7/21/2011 12:51:29 PM Pacific Daylight Time;
  + 1> whenCreated: 7/21/2011 12:18:57 PM Pacific Daylight Time;
* >> Dn: CN=NTDS Settings,CN=DC1CLONE1,CN=Servers,CN=Site1,CN=Sites,CN=Configuration,DC=contoso,DC=com
  + 1> cn: NTDS Settings;
  + 1> distinguishedName: CN=NTDS Settings,CN=DC1CLONE1,CN=Servers,CN=Site1,CN=Sites,CN=Configuration,DC=contoso,DC=com;
  + 1 > dMDLocation: CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 1> dSCorePropagationData: 0x0 = ( );
  + 3> hasMasterNCs: CN=Schema,CN=Configuration,DC=contoso,DC=com; CN=Configuration,DC=contoso,DC=com; DC=contoso,DC=com;
  + 1> instanceType: 0x4 = ( WRITE );
  + 1> invocationId: 5a2a4503-ffad-4884-a420-a0fafbc0efbe;
  + 1> msDS-Behavior-Version: 5;
  + 1> msDS-HasDomainNCs: DC=contoso,DC=com;
  + 3> msDS-HasInstantiatedNCs: B:8:0000000D:CN=Schema,CN=Configuration,DC=contoso,DC=com; B:8:0000000D:CN=Configuration,DC=contoso,DC=com; B:8:00000005:DC=contoso,DC=com;
  + 3> msDS-hasMasterNCs: N=Schema,CN=Configuration,DC=contoso,DC=com; CN=Configuration,DC=contoso,DC=com; DC=contoso,DC=com;
  + 1> name: NTDS Settings;
  + 1> objectCategory: CN=NTDS-DSA,CN=Schema,CN=Configuration,DC=contoso,DC=com;
  + 3> objectClass: top; applicationSettings; nTDSDSA;
  + 1> objectGUID: cbd9c90a-0758-4cf2-987e-fa44768ab78d;
  + 1> options: 0x1 = ( IS\_GC );
  + 1> serverReferenceBL: CN=DC1CLONE1,CN=Domain System Volume (SYSVOL share),CN=File Replication Service,CN=System,DC=contoso,DC=com;
  + 1> showInAdvancedViewOnly: TRUE;
  + 1> systemFlags: 0x2000000 = ( DISALLOW\_MOVE\_ON\_DELETE );
  + 1> uSNChanged: 17777;
  + 1> uSNCreated: 17755;
  + 1> whenChanged: 7/21/2011 12:51:29 PM Pacific Daylight Time;
  + 1> whenCreated: 7/21/2011 12:18:57 PM Pacific Daylight Time;

### IDL\_DRSWriteNgcKey (Opnum 29)

The IDL\_DRSWriteNgcKey method composes and [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) the msDS-KeyCredentialLink value on an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

1. ULONG IDL\_DRSWriteNgcKey(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_WRITENGCKEYREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_WRITENGCKEYREPLY\* pmsgOut
9. );

**hDrs**: The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion**: The version of the request message. Must be set to 1, because that is the only version supported.

**pmsgIn**: A pointer to the request message.

**pdwOutVersion**: A pointer to the version of the response message. The value must be 1 because that is the only version supported.

**pmsgOut**: A pointer to the response message.

**Return Values**: 0 if successful, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if a failure occurs.

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_WRITENGCKEYREQ

The DRS\_MSG\_WRITENGCKEYREQ union defines the request messages sent to the IDL\_DRSWriteNgcKey method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_WRITENGCKEYREQ\_V1 V1;
6. } DRS\_MSG\_WRITENGCKEYREQ;

**V1**: The version 1 request.

##### DRS\_MSG\_WRITENGCKEYREQ\_V1

The DRS\_MSG\_WRITENGCKEYREQ\_V1 structure defines a request message sent to the IDL\_DRSWriteNgcKey method.

1. typedef struct \_DRS\_MSG\_WRITENGCKEYREQ\_V1{
2. [string] const WCHAR\* pwszAccount;
3. [range(0,0xFFFF)] DWORD cNgcKey;
4. [size\_is(cNgcKey)] UCHAR \* pNgcKey;
5. } DRS\_MSG\_WRITENGCKEYREQ\_V1;

**pwszAccount**: The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to modify.

**cNgcKey**: The number of bytes in the pNgcKey array.

**pNgcKey**: The NGC key value.

##### DRS\_MSG\_WRITENGCKEYREPLY

The DRS\_MSG\_WRITENGCKEYREPLY union defines the response messages received from the IDL\_DRSWriteNgcKey method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_WRITENGCKEYREPLY\_V1 V1;
6. } DRS\_MSG\_WRITENGCKEYREPLY;

**V1**: The version 1 response.

##### DRS\_MSG\_WRITENGCKEYREPLY\_V1

The DRS\_MSG\_WRITENGCKEYREPLY\_V1 structure defines a response message received from the IDL\_DRSWriteNgcKey method.

1. typedef struct \_DRS\_MSG\_WRITENGCKEYREPLY\_V1{
2. DWORD retVal;
3. } DRS\_MSG\_WRITENGCKEYREPLY\_V1;

**retVal**: 0, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

#### Method-Specific Abstract Types and Procedures

##### AccessCheckWriteToKeyCredentialLinkAttribute

1. procedure AccessCheckWriteToKeyCredentialLinkAttribute (
2. obj: DSName,
3. newValue: boolean) : ULONG

The AccessCheckWriteToKeyCredentialLinkAttribute procedure performs an access check to determine if the client [**security context**](#gt_88d49f20-6c95-4b64-a52c-c3eca2fe5709), which MUST be retrieved using the method described in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.3.4.3, has the right to modify the msDS-KeyCredentialLink [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *obj* taking into consideration both regular and extended write property rights.

1. if AccessCheckAttr(obj,
2. msDS-KeyCredentialLink,
3. RIGHT\_DS\_WRITE\_PROPERTY) then
4. return ERROR\_SUCCESS
5. else
6. if AccessCheckAttr(obj,
7. msDS-KeyCredentialLink,
8. RIGHT\_DS\_WRITE\_PROPERTY\_EXTENDED) then
9. /\* Extended write access permits the attribute to be written \*/
10. \* provided certain constraints are met. \*/
11. isSelf: boolean
12. existingValue: boolean
13. if (!(computer in obj!objectClass))
14. return ERROR\_DS\_INSUFF\_ACCESS\_RIGHTS
15. endif
16. if (obj!ObjectSid = ClientAuthorizationInfo!UserSid)
17. isSelf := true
18. else
19. isSelf := false
20. endif
22. if (obj!msDS-KeyCredentialLink = NULL)
23. existingValue := false
24. else
25. existingValue := true
26. endif
28. if (!isSelf && newValue)
29. return ERROR\_DS\_INSUFF\_ACCESS\_RIGHTS
30. endif
32. if (newValue && existingValue)
33. return ERROR\_DS\_INSUFF\_ACCESS\_RIGHTS
34. endif
35. return ERROR\_SUCCESS
36. endif

##### ComposeKeyCredentialLinkForComputer

1. procedure ComposeKeyCredentialLinkForComputer (
2. obj: DSName,
3. keyValue: array of UCHAR) : DNBinary

The ComposeKeyCredentialLinkForComputer procedure builds a DNBinary for an msDS-KeyCredentialLink. Note that the following pseudocode uses the KEYCREDENTIALLINK\_BLOB and KEYCREDENTIALLINK\_ENTRY structures and related constants ([MS-ADTS] section 2.2.20).

1. keyDNBinary : DNBinary
2. keyBinary: array of UCHAR
3. keyBlob: KEYCREDENTIALLINK\_BLOB
4. keyEntry: KEYCREDENTIALLINK\_ENTRY
5. hashOffset: DWORD
6. hashValueOffset: DWORD
7. now: FILETIME
8. now := Current time as FILETIME
9. keyBlob!Version := KEY\_CREDENTIAL\_LINK\_VERSION\_2
10. // Write the header
11. keyBinary := keyBlob
12. // Add KeyID
13. keyEntry!Length := 32
14. keyEntry!Identifier := KeyID
15. keyEntry!Value := SHA256(keyValue)
16. keyBinary := keyBinary + keyEntry
17. // Add KeyHash
18. keyEntry!Length := 32
19. keyEntry!Identifier := KeyHash
20. keyEntry!Value := 32-byte array of UCHAR
21. // Store the location of the hash
22. hashOffset := length(keyBinary)
23. keyBinary := keyBinary + keyEntry
24. // Store the location where the hash data starts
25. hashValueOffset := length(keyBinary)
26. // Add KeyMaterial
27. keyEntry!Length = length(keyValue)
28. keyEntry!Identifier = KeyMaterial
29. keyEntry!Value = keyValue
30. keyBinary := keyBinary + keyEntry
31. // Add KeyUsage
32. keyEntry!Length := 1
33. keyEntry!Identifier := KeyUsage
34. keyEntry!Value := 1
35. keyBinary := keyBinary + keyEntry
36. // Add KeySource
37. keyEntry!Length := 1
38. keyEntry!Identifier := KeySource
39. keyEntry!Value := 0
40. keyBinary := keyBinary + keyEntry
41. // Add KeyApproximateLastLogonTimeStamp
42. keyEntry!Length := sizeof(FILETIME)
43. keyEntry!Identifier := KeyApproximateLastLogonTimeStamp
44. keyEntry!Value := now
45. keyBinary := keyBinary + keyEntry
46. // Add KeyCreationTime
47. keyEntry!Length := sizeof(FILETIME)
48. keyEntry!Identifier := KeyCreationTime
49. keyEntry!Value := now
50. keyBinary := keyBinary + keyEntry
51. // Compute and store the KeyHash value now that all subsequent fields are present
52. keyEntry := keyBinary + hashOffset
53. keyEntry!Value := SHA256(keyBinary[hashValueOffset .. (length(keyBinary)-1)])
54. keyDNBinary!DN := obj
55. keyDNBinary!Binary := keyBinary
56. return keyDNBinary

#### Server Behavior of the IDL\_DRSWriteNgcKey Method

*Informative summary of behavior*: The IDL\_DRSWriteNgcKey method sets the msDS-KeyCredentialLink [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). The IDL\_DRSWriteNgcKey method replaces any existing msDS-KeyCredentialLink attributes on the object with a new value. In this new value, the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) portion is the portion set to the account and the binary portion is set to a KEYCREDENTIALLINK\_BLOB with the following KEYCREDENTIALLINK\_ENTRY entries set. (See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 2.2.20 for structures and constants.)

* KeyID – Value set to a SHA256 hash of the pNgcKey.
* KeyHash – Value set as defined in [MS-ADTS] section 2.2.20.6.
* KeyMaterial – Value set to the pNgcKey array.
* KeyUsage - Value set to KEY\_USAGE\_NGC.
* KeySource – Value set to KEY\_SOURCE\_AD.
* KeyApproximateLastLogonTimeStamp – Value set to the current time.
* KeyCreationTime – Value set to the current time.

The effect of this method can be achieved by an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) Modify operation to the msDS-KeyCredentialLink attribute of an object.

1. ULONG IDL\_DRSWriteNgcKey(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_WRITENGCKEYREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_WRITENGCKEYREPLY\* pmsgOut);
9. accountDN: unicodestring
10. account: DSName
11. keyValue : array of UCHAR
12. err: DWORD
13. key: DNBinary
14. ValidateDRSInput(hDrs, 29)
15. pdwOutVersion^ := 1
16. pmsgOut^.V1.retVal := 0
17. /\* Input parameter validation \*/
18. if dwInVersion ≠ 1 then
19. pmsgOut^.V1.retVal := ERROR\_INVALID\_PARAMETER
20. return ERROR\_INVALID\_PARAMETER
21. endif
22. /\* Input parameter validation \*/
23. if ClientUUID(hDrs) ≠ NTDSAPI\_CLIENT\_GUID
24. pmsgOut^.V1.retVal := ERROR\_INVALID\_PARAMETER
25. return ERROR\_INVALID\_PARAMETER
26. endif
27. accountDN := pmsgIn^.V1.pwszAccount
28. keyValue := pmsgIn^.V1.pNgcKey
29. if accountDN = null or accountDN = "" then
30. pmsgOut^.V1.retVal := ERROR\_INVALID\_PARAMETER
31. return ERROR\_INVALID\_PARAMETER
32. endif
33. account := GetDSNameFromDN(accountDN);
34. if not ObjExists(account) then
35. pmsgOut^.V1.retVal := ERROR\_DS\_OBJ\_NOT\_FOUND
36. return ERROR\_DS\_OBJ\_NOT\_FOUND
37. endif
38. /\* Perform access checks \*/
39. err = AccessCheckWriteToKeyCredentialLinkAttribute(account, true)
40. if err ≠ ERROR\_SUCCESS then
41. pmsgOut^.V1.retVal := err
42. return err
43. endif
44. key := ComposeKeyCredentialLinkForComputer(account, keyValue);
45. account!msDS-KeyCredentialLink := key
46. return 0

### IDL\_DRSReadNgcKey (Opnum 30)

The IDL\_DRSReadNgcKey method reads and parses the msDS-KeyCredentialLink value on an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

1. ULONG IDL\_DRSReadNgcKey(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_READNGCKEYREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_READNGCKEYREPLY\* pmsgOut
9. );

**hDrs**: The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle returned by the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) method.

**dwInVersion**: The version of the request message. Must be set to 1, because that is the only version supported.

**pmsgIn**: A pointer to the request message.

**pdwOutVersion**: A pointer to the version of the response message. The value must be 1 because that is the only version supported.

**pmsgOut**: A pointer to the response message.

**Return Values**: 0 if successful, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if a failure occurs.

**Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)): ERROR\_INVALID\_HANDLE, ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED, and ERROR\_INVALID\_PARAMETER.

#### Method-Specific Concrete Types

##### DRS\_MSG\_READNGCKEYREQ

The DRS\_MSG\_READNGCKEYREQ union defines the request messages sent to the IDL\_DRSReadNgcKey method. Only one version, identified by *dwInVersion* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_READNGCKEYREQ\_V1 V1;
6. } DRS\_MSG\_READNGCKEYREQ;

**V1**: The version 1 request.

##### DRS\_MSG\_READNGCKEYREQ\_V1

The DRS\_MSG\_READNGCKEYREQ\_V1 structure defines a request message sent to the IDL\_DRSReadNgcKey method.

1. typedef struct \_DRS\_MSG\_READNGCKEYREQ\_V1{
2. [string] const WCHAR\* pwszAccount;
3. } DRS\_MSG\_READNGCKEYREQ\_V1;

**pwszAccount**: The [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) to read.

##### DRS\_MSG\_READNGCKEYREPLY

The DRS\_MSG\_READNGCKEYREPLY union defines the response messages received from the IDL\_DRSReadNgcKey method. Only one version, identified by *pdwOutVersion^* = 1, is currently defined.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DRS\_MSG\_READNGCKEYREPLY\_V1 V1;
6. } DRS\_MSG\_READNGCKEYREPLY;

**V1**: The version 1 response.

##### DRS\_MSG\_READNGCKEYREPLY\_V1

The DRS\_MSG\_READNGCKEYREPLY\_V1 structure defines a response message received from the IDL\_DRSReadNgcKey method.

1. typedef struct \_DRS\_MSG\_READNGCKEYREPLY\_V1{
2. DWORD retVal;
3. [range(0,0xFFFF)] DWORD cNgcKey;
4. [size\_is(cNgcKey)] UCHAR \* pNgcKey;
5. } DRS\_MSG\_READNGCKEYREPLY\_V1;

**retVal**: Zero, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

**cNgcKey**: The number of bytes in the **pNgcKey** array.

**pNgcKey**: The NGC key value.

#### Server Behavior of the IDL\_DRSReadNgcKey Method

*Informative summary of behavior*: The IDL\_DRSReadNgcKey method reads the msDS-KeyCredentialLink [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) values of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca), attempts to parses the msDS-KeyCredentialLink attribute on the object and returns the KeyMaterial field ([[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 2.2.20.6) from the first entry that is successfully parsed. The order in which the values are parsed is implementation specific. Note that the following pseudocode uses the KEYCREDENTIALLINK\_BLOB and KEYCREDENTIALLINK\_ENTRY structures and related constants ([MS-ADTS] section 2.2.20).

1. ULONG IDL\_DRSReadNgcKey(
2. [in, ref] DRS\_HANDLE hDrs,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DRS\_MSG\_READNGCKEYREQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DRS\_MSG\_READNGCKEYREPLY\* pmsgOut);
9. accountDN: unicodestring
10. account: DSName
11. keyValue : array of UCHAR
12. err: DWORD
13. key: DNBinary
14. keyBinary: array of BYTE
15. keyBlob: KEYCREDENTIALLINK\_BLOB
16. keyEntry: KEYCREDENTIALLINK\_ENTRY
17. offset: DWORD
18. ValidateDRSInput(hDrs, 30)
19. pdwOutVersion^ := 1
20. pmsgOut^.V1.retVal := 0
21. /\* Input parameter validation \*/
22. if dwInVersion ≠ 1 then
23. pmsgOut^.V1.retVal := ERROR\_INVALID\_PARAMETER
24. return ERROR\_INVALID\_PARAMETER
25. endif
26. /\* Input parameter validation \*/
27. if ClientUUID(hDrs) ≠ NTDSAPI\_CLIENT\_GUID
28. pmsgOut^.V1.retVal := ERROR\_INVALID\_PARAMETER
29. return ERROR\_INVALID\_PARAMETER
30. endif
31. accountDN := pmsgIn^.V1.pwszAccount
32. if accountDN = null or accountDN = "" then
33. pmsgOut^.V1.retVal := ERROR\_INVALID\_PARAMETER
34. return ERROR\_INVALID\_PARAMETER
35. endif
36. account := GetDSNameFromDN(accountDN);
37. if not ObjExists(account) then
38. pmsgOut^.V1.retVal := ERROR\_DS\_OBJ\_NOT\_FOUND
39. return ERROR\_DS\_OBJ\_NOT\_FOUND
40. endif
41. /\* Perform access checks \*/
42. if (!AccessCheckAttr(account,
43. msDS-KeyCredentialLink,
44. RIGHT\_DS\_READ\_PROPERTY)) then
45. return ERROR\_DS\_INSUFF\_ACCESS\_RIGHTS
46. endif
47. keyValue := NULL
48. foreach (key in obj!msDS-KeyCredentialLink)
49. keyBinary := key!Binary
50. offset := 0
51. keyBlob := keyBinary
52. offset := offset + sizeof(keyBlob)
54. if (keyBlob!Version != KEY\_CREDENTIAL\_LINK\_VERSION\_2)
55. continue
56. endif
57. while (offset < length(keyBinary))
58. keyEntry := keyBinary[offset]
59. offset :=
60. offset +
61. sizeof(keyEntry!Length) +
62. sizeof(keyEntry!Identifier) +
63. keyEntry!Length
65. if (keyEntry!Identifer = KeyMaterial)
66. keyValue := keyEntry!Value
67. break
68. endif
69. endwhile
70. if (keyValue != NULL)
71. break
72. endif
73. endfor
74. if (keyValue == NULL)
75. return ERROR\_DS\_OBJ\_NOT\_FOUND
76. endif
77. pmsgOut^.V1.pNgcKey := keyValue
78. pmsgOut^.V1.cNgcKey := length(keyValue)
79. return 0

## dsaop RPC Interface

This section specifies the methods for the dsaop [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface of the DRS Remote Protocol, in addition to the processing rules. This interface is available only when msDS-UpdateScript contains a valid value, where the validation criterion is implementation-specific.

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [IDL\_DSAPrepareScript](#Section_749197848e574cf5840f6f1bd226cf02) | Prepares the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) to run a maintenance script.  Opnum: 0 |
| [IDL\_DSAExecuteScript](#Section_1cb59761aeae4f448f9e06ae75ae45ca) | Executes a maintenance script.  Opnum: 1 |

For information on the order of method calls, see section [1.3.2](#Section_67c5a415a6c740988cf36ef8d173cfe8).

All methods MUST NOT throw exceptions.

### IDL\_DSAPrepareScript (Opnum 0)

The IDL\_DSAPrepareScript method prepares the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) to run a maintenance script.

1. ULONG IDL\_DSAPrepareScript(
2. [in] handle\_t hRpc,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DSA\_MSG\_PREPARE\_SCRIPT\_REQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DSA\_MSG\_PREPARE\_SCRIPT\_REPLY\* pmsgOut
9. );

**hRpc:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle, as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824).

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if a failure occurs.

#### Method-Specific Concrete Types

##### DSA\_MSG\_PREPARE\_SCRIPT\_REQ

The DSA\_MSG\_PREPARE\_SCRIPT\_REQ union defines the request messages sent to the [IDL\_DSAPrepareScript](#Section_749197848e574cf5840f6f1bd226cf02) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DSA\_MSG\_PREPARE\_SCRIPT\_REQ\_V1 V1;
6. } DSA\_MSG\_PREPARE\_SCRIPT\_REQ;

**V1:**  The version 1 request.

##### DSA\_MSG\_PREPARE\_SCRIPT\_REQ\_V1

The DSA\_MSG\_PREPARE\_SCRIPT\_REQ\_V1 structure defines a request message sent to the [IDL\_DSAPrepareScript](#Section_749197848e574cf5840f6f1bd226cf02) method.

1. typedef struct {
2. DWORD Reserved;
3. } DSA\_MSG\_PREPARE\_SCRIPT\_REQ\_V1;

**Reserved:**  Unused. MUST be 0 and ignored.

##### DSA\_MSG\_PREPARE\_SCRIPT\_REPLY

The DSA\_MSG\_PREPARE\_SCRIPT\_REPLY union defines the response messages received from the [IDL\_DSAPrepareScript](#Section_749197848e574cf5840f6f1bd226cf02) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DSA\_MSG\_PREPARE\_SCRIPT\_REPLY\_V1 V1;
6. } DSA\_MSG\_PREPARE\_SCRIPT\_REPLY;

**V1:**  The version 1 response.

##### DSA\_MSG\_PREPARE\_SCRIPT\_REPLY\_V1

The DSA\_MSG\_PREPARE\_SCRIPT\_REPLY\_V1 structure defines a response message received from the [IDL\_DSAPrepareScript](#Section_749197848e574cf5840f6f1bd226cf02) method.

1. typedef struct {
2. DWORD dwOperationStatus;
3. [string] LPWSTR pwErrMessage;
4. [range(0,1024)] DWORD cbPassword;
5. [size\_is(cbPassword)] BYTE\* pbPassword;
6. [range(0,10485760)] DWORD cbHashBody;
7. [size\_is(cbHashBody)] BYTE\* pbHashBody;
8. [range(0,10485760)] DWORD cbHashSignature;
9. [size\_is(cbHashSignature)] BYTE\* pbHashSignature;
10. } DSA\_MSG\_PREPARE\_SCRIPT\_REPLY\_V1;

**dwOperationStatus:**  0 if successful, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if a fatal error occurred.

**pwErrMessage:**  Null if successful, or a description of an error if a fatal error occurred.

**cbPassword:**  The count, in bytes, of the **pbPassword** array.

**pbPassword:**  The password.

**cbHashBody:**  The count, in bytes, of the **pbHashBody** array.

**pbHashBody:**  The hash of the script value.

**cbHashSignature:**  The count, in bytes, of the **pbHashSignature** array.

**pbHashSignature:**  The script signature.

#### Method-Specific Abstract Types and Procedures

##### GetKeyLengthHandleT

1. procedure GetKeyLengthHandleT(hRpc: handle\_t): integer

Returns the key length, in bits, of the encryption used on the *hRpc* connection. Returns 0 if no encryption is in use on the connection.

##### PrepareScriptInProgress

1. procedure PrepareScriptInProgress(): boolean

Returns true if an instance of the [IDL\_DSAPrepareScript](#Section_749197848e574cf5840f6f1bd226cf02)() method is already executing, and false otherwise.

##### PrepareScriptVerifyScript

1. procedure PrepareScriptVerifyScript(pc: DSName): boolean

Executes an [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) Rename(*pc*!msDS-UpdateScript, false), as specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.12. Returns true if the return value of NC Rename is 0 and false otherwise.

##### PrepareScriptHashBody

1. procedure PrepareScriptHashBody(pc: DSName): sequence of BYTE

Returns a [**SHA1 hash**](#gt_4b437f5b-b4b8-4601-9b87-3f78803638c7) of the value of *pc*!msDS-UpdateScript.

##### PrepareScriptHashSignature

1. procedure PrepareScriptHashSignature(pc: DSName): sequence of BYTE

Returns a [**SHA1 hash**](#gt_4b437f5b-b4b8-4601-9b87-3f78803638c7) of the value formed by appending the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) {0916C8E3-3431-4586-AF77-44BD3B16F961} to the value of *pc*!msDS-UpdateScript.

##### PrepareScriptGeneratePassword

1. procedure PrepareScriptGeneratePassword(): sequence of BYTE

Returns a randomly generated password for use in a subsequent call to [IDL\_DSAExecuteScript](#Section_1cb59761aeae4f448f9e06ae75ae45ca).

#### Server Behavior of the IDL\_DSAPrepareScript Method

*Informative summary of behavior*: The [IDL\_DSAPrepareScript](#Section_749197848e574cf5840f6f1bd226cf02) method prepares for a subsequent call to [IDL\_DSAExecuteScript](#Section_1cb59761aeae4f448f9e06ae75ae45ca). The [**partitions container**](#gt_d6b4c198-f9d3-4c49-b0f0-390e07f89af1) that is a [**child object**](#gt_9b04b599-9dca-48f1-aa9e-08e254d20553) of the root of the [**configuration NC**](#gt_54215750-9443-4383-866c-2a95f79f1625) is altered as follows:

* The value of msDS-UpdateScript is validated.
* If valid, a password is generated and stored in the value for msDS-ExecuteScriptPassword.

The password, a hash of the value stored in msDS-UpdateScript, and a hash of that same value with the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) {0916C8E3-3431-4586-AF77-44BD3B16F961} appended are returned to the client. The returned password value is later passed back by the client in a call to IDL\_DSAExecuteScript as a form of authorization.

1. ULONG
2. IDL\_DSAPrepareScript(
3. [in] handle\_t hRpc,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DSA\_MSG\_PREPARE\_SCRIPT\_REQ \*pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DSA\_MSG\_PREPARE\_SCRIPT\_REPLY \*pmsgOut);
10. pc: DSName
11. msgIn: DSA\_MSG\_PREPARE\_SCRIPT\_REQ\_V1
12. byteseq: sequence of BYTE
13. /\* Returned message will be version 1 \*/
14. pdwOutVersion^ := 1
15. pmsgOut^V1.dwOperationStatus := ERROR\_DS\_INTERNAL\_FAILURE
16. pmsgOut^V1.pwErrMessage := null
17. pmsgOut^V1.cbPassword := 0
18. pmsgOut^V1.pbPassword := null
19. pmsgOut^V1.cbHashBody := 0
20. pmsgOut^V1.pbHashBody := null
21. pmsgOut^V1.cbHashSignature := 0
22. pmsgOut^V1.pbHashSignature := null
23. /\* Validate the version \*/
24. if dwInVersion ≠ 1 then
25. return ERROR\_INVALID\_PARAMETER
26. endif
27. msgIn := pmsgIn^.V1
28. /\* Validate input params \*/
29. if msgIn.Reserved ≠ 0 then
30. return ERROR\_INVALID\_PARAMETER
31. endif
32. /\* Only 1 instance of this call can be running. \*/
33. if PrepareScriptInProgress() then
34. pmsgOut^.V1.dwOperationStatus := ERROR\_ACCESS\_DENIED
35. pmsgOut^.V1.pwErrMessage := human-readable description of the error
36. return 0
37. endif
38. /\* Locate the Partitions container directly beneath ConfigNC \*/
39. pc := DescendantObject(ConfigNC(), "CN=Partitions,")
40. /\* Forest functionality level must be Win2K3 or above \*/
41. if pc!msDS-Behavior-Version = null or
42. pc!msDS-Behavior-Version < DS\_BEHAVIOR\_WIN2003 then
43. return ERROR\_DS\_NOT\_SUPPORTED
44. endif
45. /\* Security checks \*/
46. if not AccessCheckAttr(
47. pc, msDS-UpdateScript, RIGHT\_DS\_WRITE\_PROPERTY) then
48. pmsgOut^.V1.dwOperationStatus := ERROR\_DS\_AUTHORIZATION\_FAILED
49. pmsgOut^.V1.pwErrMessage := human-readable description of the error
50. return 0
51. endif
52. if not AccessCheckCAR(pc, DS-Execute-Intentions-Script) then
53. pmsgOut^.V1.dwOperationStatus := ERROR\_DS\_AUTHORIZATION\_FAILED
54. pmsgOut^.V1.pwErrMessage := human-readable description of the error
55. return 0
56. endif
57. if GetKeyLengthHandleT(hRpc) < 128 then
58. pmsgOut^.V1.dwOperationStatus := ERROR\_DS\_STRONG\_AUTH\_REQUIRED
59. pmsgOut^.V1.pwErrMessage := human-readable description of the error
60. return 0
61. endif
62. /\* Validate stored script \*/
63. if not PrepareScriptVerifyScript(pc) then
64. pmsgOut^.V1.dwOperationStatus := ERROR\_DS\_INVALID\_SCRIPT
65. pmsgOut^.V1.pwErrMessage := human-readable description of the error
66. return 0
67. endif
68. /\* Generate and return password for subsequent call to
69. \* IDL\_DSAExecuteScript() \*/
70. pc!msDS-ExecuteScriptPassword := PrepareScriptGeneratePassword()
71. /\* Return password and hashes \*/
72. byteseq := pc!msDS-ExecuteScriptPassword
73. pmsgOut^.V1.pbPassword := byteseq
74. pmsgOut^.V1.cbPassword := byteseq.length
75. byteseq := PrepareScriptHashBody(pc)
76. pmsgOut^.V1.pbHashBody := byteseq
77. pmsgOut^.V1.cbHashBody := byteseq.length
78. byteseq := PrepareScriptHashSignature(pc)
79. pmsgOut^.V1.pbHashSignature := byteseq
80. pmsgOut^.V1.cbHashSignature := byteseq.length
81. pmsgOut^.V1.dwOperationStatus := 0
82. return 0

### IDL\_DSAExecuteScript (Opnum 1)

The IDL\_DSAExecuteScript method executes a maintenance script.

1. ULONG IDL\_DSAExecuteScript(
2. [in] handle\_t hRpc,
3. [in] DWORD dwInVersion,
4. [in, ref, switch\_is(dwInVersion)]
5. DSA\_MSG\_EXECUTE\_SCRIPT\_REQ\* pmsgIn,
6. [out, ref] DWORD\* pdwOutVersion,
7. [out, ref, switch\_is(\*pdwOutVersion)]
8. DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY\* pmsgOut
9. );

**hRpc:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle, as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824).

**dwInVersion:** The version of the request message.

**pmsgIn:** A pointer to the request message.

**pdwOutVersion:** A pointer to the version of the response message.

**pmsgOut:** A pointer to the response message.

**Return Values:** 0 if successful, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if a failure occurs.

#### Method-Specific Concrete Types

##### DSA\_MSG\_EXECUTE\_SCRIPT\_REQ

The DSA\_MSG\_EXECUTE\_SCRIPT\_REQ union defines the request messages sent to the [IDL\_DSAExecuteScript](#Section_1cb59761aeae4f448f9e06ae75ae45ca) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DSA\_MSG\_EXECUTE\_SCRIPT\_REQ\_V1 V1;
6. } DSA\_MSG\_EXECUTE\_SCRIPT\_REQ;

**V1:**  The version 1 request.

##### DSA\_MSG\_EXECUTE\_SCRIPT\_REQ\_V1

The DSA\_MSG\_EXECUTE\_SCRIPT\_REQ\_V1 structure defines a request message sent to the [IDL\_DSAExecuteScript](#Section_1cb59761aeae4f448f9e06ae75ae45ca) method.

1. typedef struct {
2. DWORD Flags;
3. [range(1,1024)] DWORD cbPassword;
4. [size\_is(cbPassword)] BYTE\* pbPassword;
5. } DSA\_MSG\_EXECUTE\_SCRIPT\_REQ\_V1;

**Flags:**  Unused. MUST be 0 and ignored.

**cbPassword:**  The count, in bytes, of the **pbPassword** array.

**pbPassword:**  The password.

##### DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY

The DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY union defines the response messages received from the [IDL\_DSAExecuteScript](#Section_1cb59761aeae4f448f9e06ae75ae45ca) method.

1. typedef
2. [switch\_type(DWORD)]
3. union {
4. [case(1)]
5. DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY\_V1 V1;
6. } DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY;

**V1:**  The version 1 request.

##### DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY\_V1

The DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY\_V1 structure defines a response message received from the [IDL\_DSAExecuteScript](#Section_1cb59761aeae4f448f9e06ae75ae45ca) method.

1. typedef struct {
2. DWORD dwOperationStatus;
3. [string] LPWSTR pwErrMessage;
4. } DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY\_V1;

**dwOperationStatus:**  0 if successful, or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) if a fatal error occurred.

**pwErrMessage:**  Null if successful, or a description of the error if a fatal error occurred.

#### Method-Specific Abstract Types and Procedures

##### ExecuteScriptInProgress

1. procedure ExecuteScriptInProgress(): boolean

Returns true if an instance of the [IDL\_DSAExecuteScript](#Section_1cb59761aeae4f448f9e06ae75ae45ca) method is already executing, and false otherwise.

##### ExecuteScript

1. procedure ExecuteScript(pc: DSName): ULONG

Executes an [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) Rename(*pc*!msDS-UpdateScript, true), as specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.12. Returns the return value of the NC Rename.

#### Server Behavior of the IDL\_DSAExecuteScript Method

*Informative summary of behavior*: The value of the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) msDS-UpdateScript is executed as a transacted sequence of [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493). The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) call is not authenticated using normal means (that is, it can be performed by an anonymous caller). However, the password value passed by the caller has to match the password that was obtained by a prior call to the [IDL\_DSAPrepareScript](#Section_749197848e574cf5840f6f1bd226cf02) method on the same [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

1. ULONG
2. IDL\_DSAExecuteScript(
3. [in] handle\_t hRpc,
4. [in] DWORD dwInVersion,
5. [in, ref, switch\_is(dwInVersion)]
6. DSA\_MSG\_EXECUTE\_SCRIPT\_REQ \*pmsgIn,
7. [out, ref] DWORD \*pdwOutVersion,
8. [out, ref, switch\_is(\*pdwOutVersion)]
9. DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY \*pmsgOut);
10. pc: DSName
11. msgIn: DSA\_MSG\_EXECUTE\_SCRIPT\_REQ\_V1
12. /\* returned message is version 1 \*/
13. pdwOutVersion^ := 1
14. pmsgOut^.V1.dwOperationStatus := ERROR\_DS\_INTERNAL\_FAILURE
15. pmsgOut^.V1.pwErrMessage := null
16. /\* Validate the version \*/
17. if dwInVersion ≠ 1 then
18. return ERROR\_INVALID\_PARAMETER
19. endif
20. msgIn := pmsgIn^.V1
21. /\* Only 1 instance of this call can be running. \*/
22. if ExecuteScriptInProgress() then
23. pmsgOut^.V1.dwOperationStatus := ERROR\_ACCESS\_DENIED
24. pmsgOut^.V1.pwErrMessage := human-readable description of the error
25. return 0
26. endif
27. pc := DescendantObject(ConfigNC(), "CN=Partitions,")
28. /\* Forest functionality level must be Win2K3 or above \*/
29. if pc!msDS-Behavior-Version = null or
30. pc!msDS-Behavior-Version < DS\_BEHAVIOR\_WIN2003 then
31. return ERROR\_DS\_NOT\_SUPPORTED
32. endif
33. /\* Passwords match? \*/
34. if pc!msDS-ExecuteScriptPassword ≠ msgIn.pbPassword then
35. pmsgOut^.V1.dwOperationStatus := ERROR\_DS\_AUTHORIZATION\_FAILED
36. pmsgOut^.V1.pwErrMessage := human-readable description of the error
37. return 0
38. endif
39. /\* Execute and delete the script. \*/
40. pmsgOut^.V1.dwOperationStatus := ExecuteScript(pc)
41. if pmsgOut^.V1.dwOperationStatus = 0 then
42. /\* Script executed successfully. Remove the script value \*/
43. pc!msDS-UpdateScript := null
44. else
45. pmsgOut^.V1.pwErrMessage := human-readable description of the error
46. endif
47. return 0

# Common Data Types, Variables, and Procedures

This section contains types that are used by two or more [drsuapi](#Section_58f33216d9f143bfa18387e3c899c410) or [dsaop](#Section_8988d95c631b46a4b84e16de204fb142) methods, or types that are used in this specification but normatively specified in other specifications. It also contains types and procedures used only within the specification. This section is arranged in order by type or procedure name.

The specification of message syntax in this section is normative for syntax only. The behavior descriptions for types representing messages are informative. Consult the behavior description for each method that uses a type for the normative specification of behavior related to that type.

"Hand-marshaled" types are types passed as [**BLOBs**](#gt_ad861812-8cb0-497a-80bb-13c95aa4e425) through [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) and types stored as BLOBs in the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9). Any type that is "hand-marshaled" is specified pictorially in this section to emphasize the layout of any multibyte quantities it contains. The layout is always little-endian. If a type is both "hand-marshaled" and marshaled by RPC, then an [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) specification of the type is given in addition to the pictorial specification.

This specification uses the definitions of RPC base types. Additional data types used in this protocol are specified in this section.

Note that values of some types are marshaled by RPC as structures in some cases and as little-endian byte arrays in other cases. An example is [DSName](#Section_a0d5477a522946b9890a54b924d487d1), which can be marshaled as a DSName \*pObject field of an [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b), or as a [UCHAR](#Section_aca7e264341d4231b5ac1003e89e24b7) \*pVal field of an [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4). Where such cases exist, the structure is defined both in [**MIDL**](#gt_9c5903c1-1477-4181-b451-3ba1e34a0c0c) syntax and in a byte diagram, and the byte array cases are clearly identified so that big-endian architectures can perform the necessary byte swapping. (For example, see ATTRVAL conversions.)

## AbstractPTFromConcretePT

1. procedure AbstractPTFromConcretePT(
2. concretePrefixTable: SCHEMA\_PREFIX\_TABLE): PrefixTable

*Informative summary of behavior*: The AbstractPTFromConcretePT procedure translates the [SCHEMA\_PREFIX\_TABLE](#Section_9b371267e8b84c69997902dae02e5e38) structure to an abstract [PrefixTable](#Section_2789d96b50e8444d82d6523831556d76).

1. prefixTable: PrefixTable
2. i: DWORD
3. for i := 0 to (concretePrefixTable.PrefixCount - 1)
4. prefixTable[i].prefixString :=
5. concretePrefixTable. pPrefixTableEntry[i].prefix
6. prefixTable[i].prefixIndex :=
7. concretePrefixTable.pPrefixTableEntry[i].ndx
8. endfor
9. return concretePrefixTable

## AccessCheckAttr

1. procedure AccessCheckAttr(
2. dsName: DSName, attr: ATTRTYP, right: Right): boolean

The AccessCheckAttr procedure returns true if *dsName* identifies an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) within an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) hosted by the server, and if the client's [**security context**](#gt_88d49f20-6c95-4b64-a52c-c3eca2fe5709), which MUST be retrieved using the method described in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.3.4.3, has the access indicated by the access right **right** to the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) **attr** on that object according to the algorithm specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.5.3.2. The procedure returns false otherwise.

See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 5.1.3 for the specification of this procedure.

See [MS-ADTS] section 5.1.3.2 for the list of symbolic names for access rights (for example, RIGHT\_DS\_WRITE\_PROPERTY) and the numeric value of each.

## AccessCheckCAR

1. procedure AccessCheckCAR(dsName: DSName; right: Right): boolean

The AccessCheckCAR procedure returns true if *dsName* identifies an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) within an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) hosted by the server, and if the client's [**security context**](#gt_88d49f20-6c95-4b64-a52c-c3eca2fe5709), which MUST be retrieved using the method described in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.3.4.3, has the access indicated by the [**control access right**](#gt_42f6c9e0-a2b3-4bc3-9b87-fdb902e5505e) **right** on that object according to the algorithm specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.5.3.2. It returns false otherwise.

See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 5.1.3 for the specification of this procedure.

See [MS-ADTS] section 5.1.3.2.1 for the list of symbolic names for control access rights (for example, DS-Replication-Manage-Topology) and the numeric value of each.

## AccessCheckObject

1. procedure AccessCheckObject(dsName: DSName, right: Right): boolean

The AccessCheckObject procedure returns true if *dsName* identifies an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) within an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) hosted by the server, and if the client's [**security context**](#gt_88d49f20-6c95-4b64-a52c-c3eca2fe5709), which MUST be retrieved using the method described in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.3.4.3, has the access indicated by the access right **right** on that object according to the algorithm specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.5.3.2. The procedure returns false otherwise.

See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 5.1.3 for the specification of this procedure.

See [MS-ADTS] section 5.1.3.2 for the list of symbolic names for access rights (for example, RIGHT\_DS\_DELETE\_CHILD) and the numeric value of each.

## AccessCheckWriteToSpnAttribute

1. procedure AccessCheckWriteToSpnAttribute(
2. obj: DSName, spnSet: set of unicodestring) : boolean

The AccessCheckWriteToSpnAttribute procedure performs an access check to determine if the client [**security context**](#gt_88d49f20-6c95-4b64-a52c-c3eca2fe5709), which MUST be retrieved using the method described in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.3.4.3, has the right to modify the servicePrincipalName [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *obj* with the [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) values specified in *spnSet*, taking into consideration both regular and extended write property rights.

1. if AccessCheckAttr(obj,
2. servicePrincipalName,
3. RIGHT\_DS\_WRITE\_PROPERTY) then
4. return ERROR\_SUCCESS
5. else
6. if AccessCheckAttr(obj,
7. servicePrincipalName,
8. RIGHT\_DS\_WRITE\_PROPERTY\_EXTENDED) then
9. /\* Extended write access permits the attribute to be written \*/
10. \* provided the proposed SPNs meet certain constraints. \*/
11. foreach spn in spnSet
12. if not Is2PartSPN(spn) then
13. if (Is3PartSPN(spn) and IsDCAccount(obj)) then
14. /\* Three part SPNs are permitted for DC computer accounts \*/
15. /\* However, in addition to the constraints on 2 part SPNs,\*/
16. /\* the service name must meet additional constraints \*/
17. serviceName := GetServiceNameFromSPN(spn)
18. if not IsValidServiceName(obj, serviceName)
19. return ERROR\_DS\_INVALID\_ATTRIBUTE\_SYNTAX
20. endif
21. else
22. return ERROR\_DS\_INVALID\_ATTRIBUTE\_SYNTAX
23. endif
24. endif
25. instanceName := GetInstanceNameFromSPN(spn)
26. if (instanceName ≠ obj!dNSHostName) and
27. (not instanceName + "$" = obj!sAMAccountName) and
28. (not instanceName in obj!msDS-AdditionalDnsHostName) and
29. (not instanceName + "$" in
30. obj!msDS-AdditionalSamAccountName) then
31. /\* If this is a DC computer account \*/
32. /\* the instance name might be a GUID based dns host name \*/
33. if IsDCAccount(obj) then
34. if not IsGUIDBasedDNSName(obj, instanceName)then
35. return ERROR\_DS\_INVALID\_ATTRIBUTE\_SYNTAX
36. endif
37. else
38. return ERROR\_DS\_INVALID\_ATTRIBUTE\_SYNTAX
39. endif
40. endif
41. endfor
42. return ERROR\_SUCCESS
43. endif
44. return ERROR\_DS\_INSUFF\_ACCESS\_RIGHTS
45. endif

## AddSubRef

1. procedure AddSubRef(childNC: DSName): DWORD

*Informative summary of behavior*: This procedure creates a [**sub-ref object**](#gt_a4b4bece-8452-402c-99c6-12ebf0af0b58) to the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) named by *childNC*, if appropriate.

1. parentNC, parentObj: DSName
2. srAtt: ENTINF
3. err: DWORD
4. err:= 0
5. /\* Find the parent NC \*/
6. parentNC := GetObjectNC(ChildNC)
7. /\* If the parent NC is not instantiated locally, return \*/
8. if not FullReplicaExists(parentNC) and
9. not PartialGCReplicaExists(parentNC) then
10. return err
11. endif
12. /\* If child does not exist, create it \*/
13. if !ObjExists(childNC) then
14. /\* Create a subordinate reference object \*/
15. ENTINF\_SetValue(srAtt, instanceType,
16. {IT\_NC\_HEAD | IT\_UNINSTANT | IT\_NC\_ABOVE}, dc.prefixTable)
17. ENTINF\_SetValue(srAtt, objectClass, top,prefixTable)
18. ENTINF\_SetValue(srAtt, objectCategory,SchemaObj(top), dc.prefixTable)
19. ENTINF\_SetValue(srAtt, distinguishedName, childNC,prefixTable)
20. err := PerformAddOperation(srAtt,childNC, dc.prefixTable, TRUE)
21. if(err != 0)
22. return err
23. endif
24. else
25. if (childNC!isDeleted)
26. /\*The cross-ref is being undeleted. Undelete the sub-ref object also.\*/
27. UndeleteObject(childNC)
28. endif
29. endif
30. /\* Ensure that the subordinate reference object is listed in the parent's
31. subRefs attribute\*/
32. if (not childNC in parentNC!subRefs) then
33. parentNC!subRefs := parentNC!subRefs + {childNC}
34. endif
35. return err

## AmIRODC

1. procedure AmIRODC() : boolean

The AmIRODC procedure returns true if the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) is an [**RODC**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870).

1. return DSAObj()!objectCategory = SchemaObj(nTDSDSARO)

## AmILHServer

1. procedure AmILHServer() : boolean

The AmILHServer procedure returns true if the local machine is Windows Server 2008 operating system or later.

1. /\* DS\_BEHAVIOR\_WIN2008 defined in [MS-ADTS]
2. \* section 6.1.4.2, "msDS-Behavior-Version: DC Functional Level" \*/
3. return DSAObj()!msDS-Behavior-Version ≥ DS\_BEHAVIOR\_WIN2008

## ATTR

The ATTR structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the identity and values of an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).

1. typedef struct {
2. ATTRTYP attrTyp;
3. ATTRVALBLOCK AttrVal;
4. } ATTR;

**attrTyp:**  An attribute.

**AttrVal:**  The sequence of values for this attribute.

## ATTRBLOCK

The ATTRBLOCK structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a set of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) and their values.

1. typedef struct {
2. [range(0,1048576)] ULONG attrCount;
3. [size\_is(attrCount)] ATTR\* pAttr;
4. } ATTRBLOCK;

**attrCount:**  The number of items in the pAttr array.

**pAttr:**  An array of attributes and their values.

## AttributeStamp

AttributeStamp is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that contains information about the last [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) to an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f). It is a tuple of the following:

* **dwVersion**: A 32-bit integer. Set to 1 when a value for the attribute is set for the first time. On each subsequent originating update, if the current value of **dwVersion** is less than 0xFFFFFFFF, then increment it by 1; otherwise set it to 0.
* **timeChanged**: The date and time at which the last originating update was made.
* **uuidOriginating**: The [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that performed the last originating update.
* **usnOriginating**: The [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) assigned to the last originating update by the DC that performed it.

**Comparisons**

Version Comparison: The following procedure is used for comparing the **dwVersion** fields of two **AttributeStamps**:

1. procedure CompareVersions(x: DWORD, y: DWORD): int

*Informative summary of behavior*: This procedure compares two **dwVersions** and returns an integer that is used in AttributeStamp following comparisons.

1. if x = y then
2. return 0
3. elseif x > 0x7FFFFFFF then
4. if y = (x – 0x80000000) then
5. return 1
6. elseif (y < (x – 0x7FFFFFFF)) or (x < y) then
7. return -1
8. else
9. return 1
10. endif
11. elseif x < 0x7FFFFFFF then
12. if y = (x + 0x80000000) then
13. return -1
14. elseif (x < y) and (y < (x – 0x7FFFFFFF)) then
15. return -1
16. else
17. return 1
18. endif
19. else
20. if y = 0xFFFFFFFF then
21. return -1
22. elseif x < y then
23. return -1
24. else
25. return 1
26. endif
27. endif

AttributeStamp Comparison: Given two **AttributeStamps** *x* and *y*, let *d* be the result of the procedure CompareVersions(x.dwVersion, y.dwVersion).

*x* is said to be equal to *y* if any of the following is true:

* *x* is null and *y* is null
* *d* = 0 and x.timeChanged = y.timeChanged and x.uuidOriginating = y.uuidOriginating

*x* is said to be greater than *y* if any of the following is true:

* *x* is not null and *y* is null
* *d* > 0
* *d* = 0 and x.timeChanged > y.timeChanged
* *d* = 0 and x.timeChanged = y.timeChanged and x.uuidOriginating > y.uuidOriginating

*x* is said to be less than *y* if any of the following is true:

* *x* is null and *y* is not null
* *d* < 0
* *d* = 0 and x.timeChanged < y.timeChanged
* *d* = 0 and x.timeChanged = y.timeChanged and x.uuidOriginating < y.uuidOriginating

**Conversions**

A value *x* of type **AttributeStamp** can be converted to and from its wire format *y* of type [PROPERTY\_META\_DATA\_EXT](#Section_aef7ebdec305422495fd585c86b19c38) by associating the values of fields in *x* with the values of the like-named fields in *y*.

## AttributeSyntax

AttributeSyntax is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that represents an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) [**attribute syntax**](#gt_811ea26f-62cc-472e-9aca-9448831f16d8). The valid values are the names from the LDAP Syntax Name column of the table in section [5.16.2](#Section_284c8a5a6ede4d3488babda0b8bb59e0), for example, "Object(DS-DN)" and "Object(DN-Binary)".

## AttrStamp

1. procedure AttrStamp(o: DSName, attr: ATTRTYP) : AttributeStamp

The AttrStamp procedure returns the [AttributeStamp](#Section_2973bb80c8ed450da98159639e09820b) for the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) whose [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) is *attr* on the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) whose [DSName](#Section_a0d5477a522946b9890a54b924d487d1) is *o*.

## ATTRTYP

ATTRTYP is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a compact representation of an [**OID**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

This type is declared as follows:

1. typedef ULONG ATTRTYP;

Section [5.16.4](#Section_6f53317f226348ee86c14580bf97232c) specifies the procedures that map between ATTRTYP and OID with the aid of a [SCHEMA\_PREFIX\_TABLE](#Section_9b371267e8b84c69997902dae02e5e38).

## AttrtypFromSchemaObj

1. procedure AttrtypFromSchemaObj(o: DSName): ATTRTYP

Given the [**dsname**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a) *o* of an attributeSchema or classSchema [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca), the AttrtypFromSchemaObj procedure returns the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) that identifies this [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093) object on this [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

1. if o!msDS-IntId ≠ null then
2. return o!msDS-IntId
3. endif
4. if attributeSchema in o!objectClass then
5. return MakeAttid(dc.prefixTable, o!attributeID)
6. else
7. return MakeAttid(dc.prefixTable, o!governsID)
8. endif

## ATTRVAL

The ATTRVAL structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the value of a single [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).

1. typedef struct {
2. [range(0,26214400)] ULONG valLen;
3. [size\_is(valLen)] UCHAR\* pVal;
4. } ATTRVAL;

**valLen:**  The size, in bytes, of the **pVal** array.

**pVal:**  The value of the attribute. The encoding of the attribute varies by syntax, as described in the following sections.

### Concrete Value Representations

This section defines types used for concrete value representations. In addition to the types described here, the following types are also used for concrete value representations:

* [ATTRTYP (section 5.14)](#Section_9117312908e6497c8266b5ac0aa5f983)
* [DSNAME (section 5.50)](#Section_385d478f3eb64d2cac58f25c4debdd86)
* [DSTIME (section 5.51)](#Section_a72a16b973e441caa5c1afc5fc54e175)
* [SYNTAX\_ADDRESS (section 5.192)](#Section_7df24a29d2e44f9eb55cabbd72131422)
* [SYNTAX\_DISTNAME\_BINARY (section 5.193)](#Section_8eefc5ab6d2248b4bea163b53a81a3a9)

#### INT32

The INT32 type is a 4-byte integer in little-endian form. See [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.22 for its definition.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| intValue | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**intValue (4 bytes):** A 32-bit signed number in little-endian byte order.

#### INT64

The INT64 type is an 8-byte integer in little-endian form. See [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.23 for its definition.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| int64Value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**int64Value (8 bytes):** A 64-bit signed number in little-endian byte order.

#### OctetString

The OctetString represents an array of bytes. The number of bytes in the array equals the valLen field of the [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) structure.

#### String8

The String8 type is an array of ASCII characters. Each character is encoded as a single byte. The number of bytes in the array equals the valLen field of the [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) structure.

#### String16

The String16 type is an array of [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) characters. Each Unicode character is encoded as 2 bytes. The number of bytes in the array equals the valLen field of the [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) structure.

The byte ordering is little-endian.

#### SECURITY\_DESCRIPTOR

The SECURITY\_DESCRIPTOR structure is an NT [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) in self-relative format, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.6.

The data is stored in little-endian byte order.

#### SID

The **SID** type is an NT [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) structure, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2.

The data is stored in little-endian byte order.

### Abstract Value Representations

The abstract data model utilizes a representation of data values that is used by [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d), minus the BER encoding. Several of these syntaxes are adopted from [[RFC2252]](https://go.microsoft.com/fwlink/?LinkId=90326).

The following table lists all the supported syntaxes and how they are represented in the model. Some syntaxes share an [**OID**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2), so the syntaxes in the table are identified by name, not OID.

| LDAP syntax name (OID) | [RFC2252] name | Reference section in [RFC2252] or in this document |
| --- | --- | --- |
| Boolean (2.2.5.8) | Boolean | [RFC2252] section 6.4 |
| Enumeration (2.5.5.9) | INTEGER | [RFC2252] section 6.16 |
| Integer (2.5.5.9) | INTEGER | [RFC2252] section 6.16 |
| LargeInteger (2.5.5.16) | INTEGER | [RFC2252] section 6.16 |
| Object(Presentation-Address) (2.5.5.13) | Presentation Address | [RFC2252] section 6.28 |
| Object(Replica-Link) (2.5.5.10) | Binary | [RFC2252] section 6.2 |
| String(IA5) (2.5.5.5) | IA5 String | [RFC2252] section 6.15 |
| String(Numeric) (2.5.5.6) | Numeric String | [RFC2252] section 6.23 |
| String(Object-Identifier) (2.5.5.2) | OID | [RFC2252] section 6.25 |
| String(Octet) (2.5.5.10) | Binary | [RFC2252] section 6.2 |
| String(Printable) (2.5.5.5) | Printable String | [RFC2252] section 6.29 |
| String(Unicode) (2.5.5.12) | Directory String | [RFC2252] section 6.10 |
| String(UTC-Time) (2.5.5.11) | UTC Time | [RFC2252] section 6.31 |
| String(Generalized-Time) (2.5.5.11) | Generalized Time | [RFC2252] section 6.14 |
| Object(DS-DN) (2.5.5.1) | - | Section [5.16.2.1](#Section_32896221c4cb4f2ca6a7bc2cece8fc7b) |
| Object(DN-String) (2.5.5.14) | - | Section [5.16.2.2](#Section_de1cb4d35bb04ff59da8937a7dd1134a) |
| Object(DN-Binary) (2.5.5.7) | - | Section [5.16.2.3](#Section_53e0cab126874bb1984b8063240e7430) |
| Object(Access-Point) (2.5.5.14) | - | Section [5.16.2.4](#Section_7bf069fd100640ac8707d7e7d34a1de0) |
| Object(OR-Name) (2.5.5.7) | - | Section [5.16.2.5](#Section_c030a604f05f4c6fb74259c8607e7c12) |
| String(NT-Sec-Desc) (2.5.5.15) | - | Section [5.16.2.6](#Section_97b138da550e46ee9c0217936c533852) |
| String(SID) (2.5.5.17) | - | Section [5.16.2.7](#Section_8a8b4cb8e4604611b278c17e0a2a37bb) |
| String(Teletex) (2.5.5.4) | - | Section [5.16.2.8](#Section_ebe75cecb5204de18de5cf2c00107a93) |

The LDAP syntaxes that are not defined in [RFC2252] are described in the following sections.

#### Object(DS-DN)

A value with the Object(DS-DN) syntax is a UTF-8 string in the following format:

<GUID=*guid\_value*>;<SID=*sid\_value*>;*dn*

where:

* *guid\_value* is the value of the [**object's**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) objectGUID [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).
* *sid\_value* is the value of the object's objectSid attribute in its binary format (as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2).
* *dn* is the string representation of a [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) (as specified by [[RFC2252]](https://go.microsoft.com/fwlink/?LinkId=90326) section 6.9, and further specified by [[RFC2253]](https://go.microsoft.com/fwlink/?LinkId=90327)).

For reference to objects that do not have an objectSid, the format is as follows:

<GUID=*guid\_value*>;*dn*

where *guid\_value* and *dn* have the same meaning as in the previous case.

#### Object(DN-String)

A value with the Object(DN-String) syntax is a UTF-8 string in the following format:

S:*char\_count*:*string\_value*:*object\_DN*

where:

* S is a string literal that MUST be present.
* Each : is a string literal that MUST be present.
* *char\_count* is the number of characters in the *string\_value* string.
* *object\_DN* is an [**object reference**](#gt_3ca938ae-c14f-4f59-8a7d-daca9f76db4e) in the format of [Object(DS-DN)](#Section_32896221c4cb4f2ca6a7bc2cece8fc7b).

#### Object(DN-Binary)

A value with the Object(DN-Binary) syntax is a UTF-8 string in the following format:

B:*char\_count*:*binary\_value*:*object\_DN*

where:

* B is a string literal that MUST be present.
* Each : is a string literal that MUST be present.
* *char\_count* is the number of hexadecimal digits in *binary\_value*.
* *binary\_value* is the hexadecimal representation of a binary value.
* *object\_DN* is an [**object reference**](#gt_3ca938ae-c14f-4f59-8a7d-daca9f76db4e) in the format of [Object(DS-DN)](#Section_32896221c4cb4f2ca6a7bc2cece8fc7b).

#### Object(Access-Point)

A value with the Object(Access-Point) syntax is a UTF-8 string in the following format:

*presentation\_address*#X500:*object\_DN*

where:

* #X500 is a string literal that MUST be present.
* : is a string literal that MUST be present.
* *presentation\_address* is a value encoded in the Object(Presentation-Address) syntax.
* *object\_DN* is an [**object reference**](#gt_3ca938ae-c14f-4f59-8a7d-daca9f76db4e) in the format of [Object(DS-DN)](#Section_32896221c4cb4f2ca6a7bc2cece8fc7b).

#### Object(OR-Name)

A value with the Object(OR-Name) syntax is a UTF-8 string in the following format:

*object\_DN*

where: *object\_DN* is an [**object reference**](#gt_3ca938ae-c14f-4f59-8a7d-daca9f76db4e) in the format of [Object(DS-DN)](#Section_32896221c4cb4f2ca6a7bc2cece8fc7b).

#### String(NT-Sec-Desc)

A value with the String(NT-Sec-Desc) syntax contains a Windows [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) in self-relative binary form. The binary form is that of a SECURITY\_DESCRIPTOR structure and is documented in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.6.

#### String(Sid)

A value with the String(Sid) syntax is a Windows [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) in binary form. The binary form is that of a SID structure and is specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2.

#### String(Teletex)

A value with the String(Teletex) syntax is a UTF-8 string restricted to characters with values between 0x20 and 0x7e, inclusive.

### Converting Between Abstract and Concrete Value Representations

The type [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) is an encoding that several methods use to send individual [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) values across the network. When an attribute has multiple values, and all those values need to be sent, this is performed by sending multiple ATTRVALs.

An ATTRVAL that encodes an [**OID**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2) requires a [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) for decoding. In some cases, the prefix table accompanies the ATTRVAL in the same [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) request or response. In other cases, a predefined prefix table is sufficient. The process of creating the ATTRVAL for an OID can add an entry to the prefix table that will accompany the ATTRVAL.

The abstract directory model specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1 represents individual attribute values in the form used by [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) (see [[RFC2252]](https://go.microsoft.com/fwlink/?LinkId=90326)), minus the BER encoding. In short, values are represented as strings in a variety of formats. The [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) [Value](#Section_c1b732d37bf94ba181ee07157f07294c) is used to represent an attribute value in the model. Section [5.16.2](#Section_284c8a5a6ede4d3488babda0b8bb59e0) specifies the abstract representation for each LDAP syntax.

Therefore, this specification requires procedures that convert between the concrete ATTRVAL encoding and the abstract Value encoding, creating a prefix table while creating the ATTRVAL, and reading a prefix table while decoding the ATTRVAL. These procedures have the following signatures:

1. procedure ATTRVALFromValue(
2. v: Value, s: AttributeSyntax, var t: PrefixTable) : ATTRVAL
3. procedure ValueFromATTRVAL(
4. a: ATTRVAL, s: AttributeSyntax, t: PrefixTable) : Value

where:

* *s* is an LDAP attribute syntax from the table in section 5.16.2.
* *t* is an abstract [PrefixTable](#Section_2789d96b50e8444d82d6523831556d76) [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca), representing a prefix table.

ATTRVALFromValue mutates its input PrefixTable object *t*; ValueFromATTRVAL does not.

Apart from the prefix table complication, these two procedures are straightforward given the two value representations. These procedures obey the mappings shown in the following table for converting between abstract and concrete value representations.

| LDAP syntax name | Encoding of ATTRVAL payload |
| --- | --- |
| Boolean (2.2.5.8) | INT32 |
| Enumeration (2.5.5.9) | INT32 |
| Integer (2.5.5.9) | INT32 |
| LargeInteger (2.5.5.16) | INT64 |
| Object(Presentation-Address) (2.5.5.13) | [SYNTAX\_ADDRESS](#Section_7df24a29d2e44f9eb55cabbd72131422) |
| Object(Replica-Link) (2.5.5.10) | OctetString |
| String(IA5) (2.5.5.5) | String8 |
| String(Numeric) (2.5.5.6) | String8 |
| String(Object-Identifier) (2.5.5.2) | [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) |
| String(Octet) (2.5.5.10) | OctetString |
| String(Printable) (2.5.5.5) | String8 |
| String(Unicode) (2.5.5.12) | String16 |
| String(UTC-Time) (2.5.5.11) | [DSTIME](#Section_a72a16b973e441caa5c1afc5fc54e175) |
| String(Generalized-Time) (2.5.5.11) | DSTIME |
| Object(DS-DN) (2.5.5.1) | [DSName](#Section_a0d5477a522946b9890a54b924d487d1) |
| Object(DN-String) (2.5.5.14) | [SYNTAX\_DISTNAME\_BINARY](#Section_8eefc5ab6d2248b4bea163b53a81a3a9) |
| Object(DN-Binary) (2.5.5.7) | SYNTAX\_DISTNAME\_BINARY |
| Object(Access-Point) (2.5.5.14) | SYNTAX\_DISTNAME\_BINARY |
| Object(OR-Name) (2.5.5.7) | SYNTAX\_DISTNAME\_BINARY |
| String(NT-Sec-Desc) (2.5.5.15) | SECURITY\_DESCRIPTOR |
| String(SID) (2.5.5.17) | [SID](#Section_13560cc227ff43a09d6fd686bccc5f3c) |
| String(Teletex) (2.5.5.4) | String8 |

Since the preceding procedures require a prefix table, a procedure to produce a prefix table is also required, as follows:

1. procedure NewPrefixTable() : PrefixTable

The special case value conversion between ATTRTYP and [OID](#Section_339504853a964b668a28a3a33e80302b) is provided by the following two procedures:

1. procedure MakeAttid(t: PrefixTable, o: OID) : ATTRTYP
2. procedure OidFromAttid(t: PrefixTable, attr: ATTRTYP) : OID

These three procedures, specified in section [5.16.4](#Section_6f53317f226348ee86c14580bf97232c), describe the algorithm for converting values of type OID to and from their ATTRVAL payload representation using a PrefixTable.

The conversion between an abstract Value representation and a concrete ATTRVAL representation is specified in the following sections, which are organized by abstract value type. In the examples shown:

* **LDAP Value** represents the LDAP value.
* **valLen** represents the value in the **valLen** field of the ATTRVAL structure.
* **payload** represents the data in the payload (the referent of pVal in the ATTRVAL structure).

Because prefix tables are communicated over the wire, the [ConcretePTFromAbstractPT](#Section_609de031450d4aac9c09db0ab4ccb2cb) and [AbstractPTFromConcretePT](#Section_250152d133844fd3b54e3ce141a07287) procedures are defined to convert between the abstract PrefixTable and the concrete [SCHEMA\_PREFIX\_TABLE](#Section_9b371267e8b84c69997902dae02e5e38).

#### Boolean

The Boolean [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) value FALSE corresponds to an INT32 with value 0. The Boolean LDAP attribute value TRUE corresponds to an INT32 with a nonzero value. INT32 is in little-endian format. The valLen field of [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) equals 4.

Example:

1. LDAP value: TRUE
2. INT32 value 0x1
3. valLen: 4.
4. payload:
5. 01 00 00 00 ....

#### Enumeration and Integer

The Enumeration and the Integer [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) syntax types are represented in the same manner. The LDAP representation of the integer as a string expressed in base-10 notation corresponds to an [INT32](#Section_4160557351bc49c2abcdd8d4c31ab1f5), which is in little-endian format. The valLen field of [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) equals 4.

Example:

1. LDAP value: 5
2. INT32 value: 0x5
3. valLen: 4,
4. payload:
5. 05 00 00 00 ....

#### LargeInteger

The [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) representation of the integer as a string expressed in base-10 notation corresponds to an [INT64](#Section_1c3855efb0584248866f70aa740b5a7b), which is in little-endian format. The valLen field of [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) equals 8.

Example:

1. LDAP value: 12605
2. INT64 value: Hexadecimal 0x313d
3. valLen: 8,
4. payload:
5. 3d 31 00 00 00 00 00 00 =1......

#### Object(Presentation-Address)

To convert from the [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) representation to the [SYNTAX\_ADDRESS](#Section_7df24a29d2e44f9eb55cabbd72131422) representation, the UTF-8 encoded string is converted to a UCS-16 encoded [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string. The resulting string is not null-terminated. The dataLen field of SYNTAX\_ADDRESS equals the length of the Unicode string in bytes, plus 4. The valLen field of [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) equals the dataLen field of SYNTAX\_ADDRESS.

Example:

1. LDAP value: 12345 (Unicode string encoded as UTF-8)
2. This represents the following SYNTAX\_ADDRESS struct:
3. +0x000 dataLen : 0xe
4. +0x004 uVal : L"12345"
5. valLen: 14
6. payload:
7. 0e 00 00 00 31 00 32 00 33 00 34 00 35 00 ....1.2.3.4.5.

#### Object(Replica-Link) String (Octet)

The representation used in [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) syntax and encoding of the [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) payload is the same. Therefore, the payload is set to the same value. The valLen field of ATTRVAL equals the length of the byte array.

#### String(IA5) String(Printable) String(Numeric) String(Teletex)

The representation used in [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) syntax and encoding of the [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) payload is the same. Therefore, the payload is set to the same value. The string is not null-terminated. The valLen field of ATTRVAL equals the number of bytes in the string.

Example:

1. LDAP value: 123456789
2. This represents an ASCII string "123456789"

valLen: 9

1. payload:
2. 31 32 33 34 35 36 37 38 39 123456789

#### String(Unicode)

To convert from the [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) representation to the [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) syntax representation, the UTF-8 encoded string is converted to a UCS-16 encoded String16. Each Unicode character is in little-endian format. The resulting string is not null-terminated. The valLength field of [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) equals the total number of bytes in the Unicode string.

Example:

1. LDAP value: Administrator (Unicode string encoded in UTF-8)
2. valLen: 26
3. payload:
4. 41 00 64 00 6d 00 69 00 6e 00 69 00 73 00 74 00 A.d.m.i.n.i.s.t.
5. 72 00 61 00 74 00 6f 00 72 00 r.a.t.o.r.

#### String(Object-Identifier)

Conversion from the [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) representation to [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) is performed via the [MakeAttid](#Section_3f7127da084a4cb2ad1349871a733c91) function. The length of the valLen field in [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) equals 4.

Conversion from ATTRTYP to LDAP representation is performed by the [OidFromAttid](#Section_fc3acd51af2a41e085977037e01454cc) procedure.

Example:

1. LDAP value: 2.5.6.5
2. This corresponds to ATTRTYP value: 0x00010005.
3. valLen: 4
4. payload:
5. 05 00 01 00 ....

#### String(UTC-Time) and String(Generalized-Time)

For both the String(UTC-Time) and String(Generalized-Time) [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) syntaxes, the time expressed in the LDAP value corresponds to [DSTIME](#Section_a72a16b973e441caa5c1afc5fc54e175). It is in little-endian format. The valLen field of [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) equals 8.

1. LDAP value: 20060609211106.0Z (06/09/2006 14:11:06 PST).
2. This corresponds to DSTIME value: 0x2fa9a74ea
3. valLen: 8,
4. payload:
5. ea 74 9a fa 02 00 00 00 .t......

#### Object(DS-DN)

The [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) representation of Object(DS-DN) is defined in section [5.16.2.1](#Section_32896221c4cb4f2ca6a7bc2cece8fc7b). This corresponds to [DSName](#Section_a0d5477a522946b9890a54b924d487d1) as follows:

The *dn* part of the LDAP representation is converted to a UCS-16 encoded [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string. Then, the attributeValue component (defined in [[RFC2253]](https://go.microsoft.com/fwlink/?LinkId=90327)) of each [**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) in the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) is canonicalized according to the following rules:

* The first leading space, if any, is escaped as a backslash (\) followed by a space.
* Any carriage return or line-feed characters are escaped as a backslash followed by the 2-digit hexadecimal value of that character, as specified in [RFC2253] section 2.4.
* Any of the following characters—number sign (#), plus sign (+), comma (,), semicolon (;), quotation mark ("), left angle bracket (<), equal sign (=), right angle bracket (>), and backslash (\)—are escaped as a backslash followed by the character.
* The trailing space, if any, is escaped as a backslash followed by a space.

The resulting string (including a terminating null character) is inserted into the StringName field of the [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86). The length of the string, in Unicode characters, is inserted into the NameLen field. The length of the string in the NameLen field does not include the terminating null character. The value of *guid\_value* in LDAP representation is expressed as a [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) and inserted into the Guid field of the DSNAME structure. If the *sid\_value* is present, it is copied into the Sid field of the DSNAME and the SidLen field is set to the length, in bytes, of the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d). If the *sid\_value* part is not present, then the SidLen field is set to 0. The valLen field of [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) equals the length of the DSNAME structure. All the multibyte quantities in the DSNAME are stored in little-endian format.

Example:

1. LDAP Value: <GUID=3ceab4a1-fc47-4a71-8195-454faa6423a3>; <SID=01050000000000051500000089598d33d3c56b6894e1f2e6f4010000>;CN=Administrator,OU=Users,DC=test,DC=com
2. This corresponds to the following DSNAME:
3. +0x000 structLen : 0x8a
4. +0x004 SidLen : 0x1c
5. +0x008 Guid : 3ceab4a1-fc47-4a71-8195-454faa6423a3
6. +0x018 Sid : S-1-5-21-864901513-1751893459-3874677140-500
7. +0x034 NameLen : 0x28
8. +0x038 StringName : L"CN=Administrator,OU=Users,DC=test,DC=com"
9. valLen: 138,
10. payload:
11. 8a 00 00 00 1c 00 00 00 a1 b4 ea 3c 47 fc 71 4a ...........<G.qJ
12. 81 95 45 4f aa 64 23 a3 01 05 00 00 00 00 00 05 ..EO.d#.........
13. 15 00 00 00 89 59 8d 33 d3 c5 6b 68 94 e1 f2 e6 .....Y.3..kh....
14. f4 01 00 00 28 00 00 00 43 00 4e 00 3d 00 41 00 ....(...C.N.=.A.
15. 64 00 6d 00 69 00 6e 00 69 00 73 00 74 00 72 00 d.m.i.n.i.s.t.r.
16. 61 00 74 00 6f 00 72 00 2c 00 4f 00 55 00 3d 00 a.t.o.r.,.O.U.=.
17. 55 00 73 00 65 00 72 00 73 00 2c 00 44 00 43 00 U.s.e.r.s.,.D.C.
18. 3d 00 74 00 65 00 73 00 74 00 2c 00 44 00 43 00 =.t.e.s.t.,.D.C.
19. 3d 00 63 00 6f 00 6d 00 00 00 =.c.o.m...

#### Object(DN-Binary)

The [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) representation of the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) value corresponds to [SYNTAX\_DISTNAME\_BINARY](#Section_8eefc5ab6d2248b4bea163b53a81a3a9). The *object\_DN* portion of the LDAP representation is treated as if it were in Object(DS-DN) syntax and converted to the [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) syntax representation, as explained in section [5.16.2.3](#Section_53e0cab126874bb1984b8063240e7430). The *binary\_value* portion of the LDAP representation is converted to the binary value (an array of bytes) and stored in the byteVal field of the [SYNTAX\_ADDRESS](#Section_7df24a29d2e44f9eb55cabbd72131422) structure. The dataLen field of SYNTAX\_ADDRESS is set to the length of the array, in bytes, plus 4, where 4 is the length of the dataLen field.

Padding is added between the DSNAME and SYNTAX\_ADDRESS structures so that the length of DSNAME plus padding modulo 4 equals 0. The padding is an array of bytes, each byte of value 0. The valLen field of [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) equals the length of the DSNAME structure, plus the number of bytes added for padding, plus the length of the SYNTAX\_ADDRESS structure.

All the multibyte quantities in the DSNAME and SYNTAX\_ADDRESS structures are stored in little-endian format.

Example where padding is required because DSNAME is not 4-byte aligned:

1. LDAP value:
2. B:8:00000005:<GUID=2d8b0ce6-aa32-4f31-a6e8-88343e6244a5>;<SID=010100001cd509a018459359>;DC=test,DC=com
3. Representation of data as SYNTAX\_DISTNAME\_BINARY:
4. +0x000 Name : DSNAME
5. +0x000 structLen : 0x56
6. +0x004 SidLen : 0xc
7. +0x008 Guid : 2d8b0ce6-aa32-4f31-a6e8-88343e6244a5
8. +0x018 Sid : S-1-483723680-1502823704
9. +0x034 NameLen : 0xe
10. +0x038 StringName : "DC=test,DC=com"
11. +0x058 Data : SYNTAX\_ADDRESS
12. +0x000 dataLen : 8
13. +0x004 byteVal : 00 00 00 05
14. valLength: 96
15. payload:
16. 56 00 00 00 0c 00 00 00 e6 0c 8b 2d 32 aa 31 4f V..........-2.1O
17. a6 e8 88 34 3e 62 44 a5 01 01 00 00 1c d5 09 a0 ...4>bD.........
18. 18 45 93 59 00 00 00 00 00 00 00 00 00 00 00 00 .E.Y............
19. 00 00 00 00 0e 00 00 00 44 00 43 00 3d 00 74 00 ........D.C.=.t.
20. 65 00 73 00 74 00 2c 00 44 00 43 00 3d 00 63 00 e.s.t.,.D.C.=.c.
21. 6f 00 6d 00 00 00 00 00 08 00 00 00 00 00 00 05 o.m.............

Example where padding is not required because DSNAME is 4-byte aligned:

1. LDAP value:
2. B:8:0000000D:<GUID= ff432fe0-8c94-43cf-915c-286b197b0164>;<SID=010100001a180dba5ec27614>;DC=test1,DC=test,DC=com.
3. Representation of data as SYNTAX\_DISTNAME\_BINARY:
4. +0x000 Name : DSNAME
5. +0x000 structLen : 0x68
6. +0x004 SidLen : 0xc
7. +0x008 Guid : ff432fe0-8c94-43cf-915c-286b197b0164
8. +0x018 Sid : S-1-437783994-343327326
9. +0x034 NameLen : 0x17
10. +0x038 StringName : "DC=test1,DC=test,DC=com"
11. +0x068 Data : SYNTAX\_ADDRESS
12. +0x000 dataLen : 0x74003d
13. +0x004 byteVal : 00 00 00 0d
14. 68 00 00 00 0c 00 00 00 e0 2f 43 ff 94 8c cf 43 h......../C....C
15. 91 5c 28 6b 19 7b 01 64 01 01 00 00 1a 18 0d ba .\(k.{.d........
16. 5e c2 76 14 00 00 00 00 00 00 00 00 00 00 00 00 ^.v.............
17. 00 00 00 00 17 00 00 00 44 00 43 00 3d 00 74 00 ........D.C.=.t.
18. 65 00 73 00 74 00 31 00 2c 00 44 00 43 00 3d 00 e.s.t.1.,.D.C.=.
19. 74 00 65 00 73 00 74 00 2c 00 44 00 43 00 3d 00 t.e.s.t.,.D.C.=.
20. 63 00 6f 00 6d 00 00 00 08 00 00 00 00 00 00 0d c.o.m...........

#### Object(DN-String)

The [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) representation of the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) value corresponds to [SYNTAX\_DISTNAME\_BINARY](#Section_8eefc5ab6d2248b4bea163b53a81a3a9). The *object\_DN* portion of the LDAP representation is treated as if it were in Object(DS-DN) syntax and converted to the [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) representation, as explained in section [5.16.2.2](#Section_de1cb4d35bb04ff59da8937a7dd1134a). The result is stored in the Name field of that structure. The *string\_value* portion of the LDAP representation is converted to a UCS-16 encoded [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string and stored in the byteVal field of the [SYNTAX\_ADDRESS](#Section_7df24a29d2e44f9eb55cabbd72131422) structure. The dataLen field of SYNTAX\_ADDRESS is set to the length of the string, in bytes, plus 4, where 4 is the length of the dataLen field. Enough padding is added between the DSNAME and SYNTAX\_ADDRESS structures such that the length of DSNAME plus padding modulo 4 equals 0. The padding is an array of bytes of value 0. The valLen field of [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) equals the length of the DSNAME structure, plus the number of bytes added for padding, plus the length of the SYNTAX\_ADDRESS structure. All the multibyte quantities in the DSNAME and SYNTAX\_ADDRESS structures are stored in little-endian format.

Example:

1. LDAP value:
2. S:7:Unicode:<GUID=2d8b0ce6-aa32-4f31-a6e8-88343e6244a5>;<SID=010100001cd509a018459359>;DC=test,DC=com
3. This represents data as SYNTAX\_DISTNAME\_BINARY
4. (note the structure SYNTAX\_ADDRESS is 4-byte aligned):
5. +0x000 Name : DSNAME
6. +0x000 structLen : 0x56
7. +0x004 SidLen : 0xc
8. +0x008 Guid : 2d8b0ce6-aa32-4f31-a6e8-88343e6244a5
9. +0x018 Sid : S-1-483723680-1502823704
10. +0x034 NameLen : 0xe
11. +0x038 StringName : "DC=test,DC=com"
12. +0x058 Data : SYNTAX\_ADDRESS
13. +0x000 dataLen : 0x12
14. +0x004 uVal : "Unicode"
15. valLength: 106
16. payload:
17. 56 00 00 00 0c 00 00 00 e6 0c 8b 2d 32 aa 31 4f V..........-2.1O
18. a6 e8 88 34 3e 62 44 a5 01 01 00 00 1c d5 09 a0 ...4>bD.........
19. 18 45 93 59 00 00 00 00 00 00 00 00 00 00 00 00 .E.Y............
20. 00 00 00 00 0e 00 00 00 44 00 43 00 3d 00 74 00 ........D.C.=.t.
21. 65 00 73 00 74 00 2c 00 44 00 43 00 3d 00 63 00 e.s.t.,.D.C.=.c.
22. 6f 00 6d 00 00 00 00 00 12 00 00 00 55 00 6e 00 o.m.........U.n.
23. 69 00 63 00 6f 00 64 00 65 00 i.c.o.d.e.

#### Object(OR-Name)

The [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) representation of the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) value corresponds to [SYNTAX\_DISTNAME\_BINARY](#Section_8eefc5ab6d2248b4bea163b53a81a3a9). The *object\_DN* of the LDAP representation is treated as if it were in Object(DS-DN) syntax and converted to the [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) syntax representation, as explained in section [5.16.2.5](#Section_c030a604f05f4c6fb74259c8607e7c12).

#### Object(Access-Point)

The [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) representation of the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) value corresponds to [SYNTAX\_DISTNAME\_BINARY](#Section_8eefc5ab6d2248b4bea163b53a81a3a9). The *object\_DN* portion of the LDAP representation is treated as if it were in Object(DS-DN) syntax and converted to the [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) syntax representation, as explained in section [5.16.2.4](#Section_7bf069fd100640ac8707d7e7d34a1de0). The *presentation\_address* portion of the LDAP representation is treated as if it were in the Object(Presentation-Address) syntax and converted to the [SYNTAX\_ADDRESS](#Section_7df24a29d2e44f9eb55cabbd72131422) representation. All the multibyte quantities in the DSNAME and SYNTAX\_ADDRESS structures are stored in little-endian format.

#### String(Sid)

The representation used in [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) syntax and encoding of [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) payload is the same. Therefore the payload is set to the same value. The valLen field of ATTRVAL equals the number of bytes in the payload. It is always 28. All the multibyte quantities in the [SID](#Section_13560cc227ff43a09d6fd686bccc5f3c) structure are stored in little-endian format.

Example:

1. LDAP Value: 01050000000000051500000089598d33d3c56b6894e1f2e6f4010000
2. valLen: 28
3. payLoad:
4. 01 05 00 00 00 00 00 05 15 00 00 00 89 59 8d 33 .............Y.3
5. d3 c5 6b 68 94 e1 f2 e6 f4 01 00 00 ..kh........

#### String(NT-Sec-Desc)

The representation used in [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) syntax and encoding of [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) payload is the same. Therefore the payload is set to the same value. The valLen field of ATTRVAL equals the number of bytes in the payload. All the multibyte quantities in the [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) structure are stored in little-endian format.

1. LDAP value: (binary blob, represented in hex format here)
2. 0100048c7000000080000000000000001400000004005c0003000000050028000001000001000000531a72ab2f1ed011981900aa0040529b01010000000000050a00000000121800ff010f0001020000000000052000000020020000001214009400020001010000000000050b000000010200001cd509a01845935900020000010200001cd509a01845935900020000
3. This represents the following self-relative security descriptor
4. value:
5. SD Revision: 1
6. SD Control: 0x8c04
7. SE\_DACL\_PRESENT
8. SE\_DACL\_AUTO\_INHERITED
9. SE\_SACL\_AUTO\_INHERITED
10. SE\_SELF\_RELATIVE
11. Owner: S-1-483723680-1502823704-512
12. Group: S-1-483723680-1502823704-512
13. DACL:
14. Revision 4
15. Size: 92 bytes
16. # Aces: 3
17. Ace[0]
18. Ace Type: 0x5 - ACCESS\_ALLOWED\_OBJECT\_ACE\_TYPE
19. Ace Size: 40 bytes
20. Ace Flags: 0x0
21. Object Ace Mask: 0x00000100
22. ACTRL\_DS\_CONTROL\_ACCESS
23. Object Ace Flags: 0x1
24. ACE\_OBJECT\_TYPE\_PRESENT
25. Object Ace Type:
26. Change Password-ab721a53-1e2f-11d0-9819-00aa0040529b
27. Object Ace Sid: NT AUTHORITY\SELF [S-1-5-10]
28. Ace[1]
29. Ace Type: 0x0 - ACCESS\_ALLOWED\_ACE\_TYPE
30. Ace Size: 24 bytes
31. Ace Flags: 0x12
32. CONTAINER\_INHERIT\_ACE
33. INHERITED\_ACE
34. Ace Mask: 0x000f01ff
35. DELETE
36. READ\_CONTROL
37. WRITE\_DAC
38. WRITE\_OWNER
39. ACTRL\_DS\_CREATE\_CHILD
40. ACTRL\_DS\_DELETE\_CHILD
41. ACTRL\_DS\_LIST
42. ACTRL\_DS\_SELF
43. ACTRL\_DS\_READ\_PROP
44. ACTRL\_DS\_WRITE\_PROP
45. ACTRL\_DS\_DELETE\_TREE
46. ACTRL\_DS\_LIST\_OBJECT
47. ACTRL\_DS\_CONTROL\_ACCESS
48. Ace Sid: BUILTIN\Administrators [S-1-5-32-544]
49. Ace[2]
50. Ace Type: 0x0 - ACCESS\_ALLOWED\_ACE\_TYPE
51. Ace Size: 20 bytes
52. Ace Flags: 0x12
53. CONTAINER\_INHERIT\_ACE
54. INHERITED\_ACE
55. Ace Mask: 0x00020094
56. READ\_CONTROL
57. ACTRL\_DS\_LIST
58. ACTRL\_DS\_READ\_PROP
59. ACTRL\_DS\_LIST\_OBJECT
60. Ace Sid: NT AUTHORITY\Authenticated Users [S-1-5-11]
61. valLen: 144
62. paylaod:
63. 01 00 04 8c 70 00 00 00 80 00 00 00 00 00 00 00 ....p...........
64. 14 00 00 00 04 00 5c 00 03 00 00 00 05 00 28 00 ......\.......(.
65. 00 01 00 00 01 00 00 00 53 1a 72 ab 2f 1e d0 11 ........S.r./...
66. 98 19 00 aa 00 40 52 9b 01 01 00 00 00 00 00 05 .....@R.........
67. 0a 00 00 00 00 12 18 00 ff 01 0f 00 01 02 00 00 ................
68. 00 00 00 05 20 00 00 00 20 02 00 00 00 12 14 00 .... ... .......
69. 94 00 02 00 01 01 00 00 00 00 00 05 0b 00 00 00 ................
70. 01 02 00 00 1c d5 09 a0 18 45 93 59 00 02 00 00 .........E.Y....
71. 01 02 00 00 1c d5 09 a0 18 45 93 59 00 02 00 00 .........E.Y....

### ATTRTYP-to-OID Conversion

This section describes the prefix mapping mechanism that allows the one-to-one mapping between [**OIDs**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2) and a 32-bit integer ([ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983)).

An OID can be represented in the binary form, with a BER encoding scheme. The standard BER encoding of an object identifier consists of three components, because the end-of-contents component is not present. Only the third component (contents octets) is used here; other components are omitted.

**Note**  The BER encoding of an OID is described in [[ITUX690]](https://go.microsoft.com/fwlink/?LinkId=89924) section 8.19. To avoid ambiguity, the non-encoded form of the OID is referred to as the original form in this section.

The prefix of an OID is the [**binary OID**](#gt_e536be2f-ccce-4557-8525-1cf9ebf2bc78), excluding the last one or two bytes. If the number following the final period (.) in the original form of the OID is less than 128, only the last byte is excluded; otherwise, the last two bytes are excluded.

A **PrefixTable** is a sequence of tuples defined as follows.

1. type PrefixTable = sequence of [
2. prefixString: unicodestring,
3. prefixIndex: integer
4. ]

where:

* *prefixString* is the prefix of an OID.
* *prefixIndex* is an integer in the range [0 .. 0x0000ffff].

The integer *prefixIndex* is called the prefix index of *prefixString*. To allow one-to-one mappings between the prefix strings and the prefix indexes in the table, each *prefixString* MUST occur at most once in the table, and each *prefixIndex* MUST occur at most once in the table.

An ATTRTYP is a 32-bit, unsigned integer. If *attr* is an ATTRTYP, define attr.upperWord to be the most significant 16 bits, and attr.lowerWord to be the least significant 16 bits.

The following types and helper procedures are used for mapping between OIDs and ATTRTYP.

1. procedure ToBinary(st: unicodestring) : sequence of BYTE

Converts a string to a binary OID representation. For example, "\x55\x06" is the binary OID \x55\x06.

1. procedure CatBinary(o: sequence of BYTE, b: BYTE) : sequence of BYTE

Concatenates a byte onto a binary OID. For example, \x02 concatenated onto \x55\x06 is \x55\x06\x02.

1. procedure ToStringOID(o: sequence of BYTE) : unicodestring

Converts a binary OID to its string representation, as described in [ITUX690] section 8.19; returns null if the conversion fails. For example, the binary OID \x55\x06\x02 is converted to the OID string "2.5.6.2".

1. procedure ToBinaryOID(s: unicodestring) : sequence of BYTE

Converts an OID string representation to a binary OID, as described in [ITUX690] section 8.19; returns null if the conversion fails. For example, the OID string "2.5.6.2" is converted to the binary form \x55\x06\x02.

1. procedure ToByte(i: integer) : BYTE

Converts an integer into a byte representation, truncating to the least significant digits, if needed. For example, 2 converts to \x02.

1. procedure SubBinary(b: sequence of BYTE,
2. start: integer, end: integer) : sequence of BYTE

Returns the sequence [*start* .. *end*] of bytes in *b*.

1. procedure AddPrefixTableEntry(var t: PrefixTable, o: sequence of BYTE)

Sets *t*[*t*.length].prefixString to *o*. Generates a random number between 0 and 65535 that is unique in the values of *prefixIndex* in *t*, and sets *t*[*t*.length].prefixIndex to the generated random number. Increases *t*.length by one.

1. procedure ToInteger(s: unicodestring) : integer

Converts a string to its integer representation. For example, "127" is 127. Strings with non-numeric characters are not defined for this procedure.

The following procedures are used for mapping between object identifiers and ATTRTYP representations.

1. procedure MakeAttid(var t: PrefixTable, o: OID): ATTRTYP

*Informative summary of behavior*: This procedure converts an OID to a corresponding ATTRTYP representation.

1. lastValueString: unicodestring
2. lastValue, lowerWord: integer
3. binaryOID, oidPrefix: sequence of BYTE
4. attr: ATTRTYP
5. pos: integer
6. /\* get the last value in the original OID: the value
7. \* after the last '.'\*/
8. lastValueString := SubString(o,
9. FindCharRev(o, o.length,'.'),
10. o.length)
11. lastValue := ToInteger(lastValueString)
12. /\* convert the dotted form of OID into a BER encoded binary
13. \* format. The BER encoding of OID is described in section
14. \* 8.19 of [ITUX690]\*/
15. binaryOID := ToBinaryOid(o)
16. /\* get the prefix of the OID\*/
17. if lastValue < 128 then
18. oidPrefix := SubBinary(binaryOID, 0, binaryOID.length - 2)
19. else
20. oidPrefix := SubBinary(binaryOID, 0, binaryOID.length - 3)
21. endif
22. /\* search the prefix in the prefix table, if none found, add
23. \* one entry for the new prefix.\*/
24. fToAdd := true
25. for i := 0 to t.length
26. if ToBinary(t[i].prefixString) = oidPrefix then
27. fToAdd := false
28. pos := i
29. endif
30. endfor
31. if fToAdd then
32. pos := t.length
33. AddPrefixTableEntry(t, oidPrefix)
34. endif
35. /\*compose the attid\*/
36. lowerWord := lastValue mod 16384
37. if lastValue ≥ 16384 then
38. /\*mark it so that it is known to not be the whole lastValue\*/
39. lowerWord := lowerWord + 32768
40. endif
41. upperWord := t[pos].prefixIndex
42. attr := upperWord \* 65536 + lowerWord
43. return attr
44. procedure OidFromAttid(t: PrefixTable, attr: ATTRTYP): OID

*Informative summary of behavior*: This procedure converts an ATTRTYP representation to a corresponding OID.

1. i, upperWord, lowerWord: integer
2. binaryOID: sequence of BYTE
3. binaryOID = null
4. /\* separate the ATTRTYP into two parts\*/
5. upperWord := attr / 65536
6. lowerWord := attr mod 65536
7. /\* search in the prefix table to find the upperWord, if found,
8. \* construct the binary OID by appending lowerWord to the end of
9. \* found prefix.\*/
10. for i := 0 to t.length
11. if t[i].prefixIndex = upperWord then
12. if lowerWord < 128 then
13. binaryOID := CatBinary(ToBinary(t[i].prefixString),
14. ToByte(lowerWord))
15. else
16. if lowerWord ≥ 32768 then
17. lowerWord := lowerWord - 32768
18. endif
19. binaryOID := CatBinary(ToBinary(t[i].prefixString),
20. ToByte(((lowerWord / 128) mod 128) + 128))
21. binaryOID := CatBinary(binaryOID, ToByte(lowerWord mod 128))
22. endif
23. endif
24. endfor
25. if binaryOID = null then
26. return null
27. else
28. return ToStringOID(binaryOID)
29. endif
30. procedure NewPrefixTable( ): PrefixTable

This procedure creates a new **PrefixTable**, inserts the following tuples into the table, and returns the table as the result.

| prefixString | Length of prefixString | prefixIndex |
| --- | --- | --- |
| "\x55\x4" | 2 | 0 |
| "\x55\x6" | 2 | 1 |
| "\x2A\x86\x48\x86\xF7\x14\x01\x02" | 8 | 2 |
| "\x2A\x86\x48\x86\xF7\x14\x01\x03" | 8 | 3 |
| "\x60\x86\x48\x01\x65\x02\x02\x01" | 8 | 4 |
| "\x60\x86\x48\x01\x65\x02\x02\x03" | 8 | 5 |
| "\x60\x86\x48\x01\x65\x02\x01\x05" | 8 | 6 |
| "\x60\x86\x48\x01\x65\x02\x01\x04" | 8 | 7 |
| "\x55\x5" | 2 | 8 |
| "\x2A\x86\x48\x86\xF7\x14\x01\x04" | 8 | 9 |
| "\x2A\x86\x48\x86\xF7\x14\x01\x05" | 8 | 10 |
| "\x09\x92\x26\x89\x93\xF2\x2C\x64" | 8 | 19 |
| "\x60\x86\x48\x01\x86\xF8\x42\x03" | 8 | 20 |
| "\x09\x92\x26\x89\x93\xF2\x2C\x64\x01" | 9 | 21 |
| "\x60\x86\x48\x01\x86\xF8\x42\x03\x01" | 9 | 22 |
| "\x2A\x86\x48\x86\xF7\x14\x01\x05\xB6\x58" | 10 | 23 |
| "\x55\x15" | 2 | 24 |
| "\x55\x12" | 2 | 25 |
| "\x55\x14" | 2 | 26 |

The following examples show the correspondence between [OID](#Section_339504853a964b668a28a3a33e80302b) and ATTRTYP by using the **PrefixTable** returned by the procedure [NewPrefixTable](#Section_e1830191fedd422690de014d0fa73a26).

1. OID: 2.5.4.6 (countryName attribute)
2. Binary: \x55\x04\x06
3. Prefix string: "\x55\x04"
4. Prefix index: 0
5. ATTRTYP: 0x00000006
6. OID: 2.5.6.2 (country class)
7. Binary: \x55\x06\x02
8. Prefix string: "\x55\x06"
9. Prefix index: 1
10. ATTRTYP: 0x00010002
11. OID: 1.2.840.113556.1.2.1 (instanceType attribute)
12. Binary: \x2A\x86\x48\x86\xF7\x14\x01\x02\x01
13. Prefix string: "\x2A\x86\x48\x86\xF7\x14\x01\x02"
14. Prefix index: 2
15. ATTRTYP: 0x00020001
16. OID: 1.2.840.113556.1.3.23 (container class)
17. Binary: \x2A\x86\x48\x86\xF7\x14\x01\x03\x17
18. Prefix string: "\x2A\x86\x48\x86\xF7\x14\x01\x03"
19. Prefix index: 3
20. ATTRTYP: 0x00030017
21. OID: 2.5.5.1 (attribute syntax: distinguished name)
22. Binary: \x55\x5\x1
23. Prefix string: "\x55\x5"
24. Prefix index: 8
25. ATTRTYP: 0x00080001
26. OID: 1.2.840.113556.1.4.1 (RDN attribute)
27. Binary: \x2A\x86\x48\x86\xF7\x14\x01\x04\x01
28. Prefix string: "\x2A\x86\x48\x86\xF7\x14\x01\x04"
29. Prefix index: 9
30. ATTRTYP: 0x00090001
31. OID: 1.2.840.113556.1.5.1 (securityObject class)
32. Binary: \x2A\x86\x48\x86\xF7\x14\x01\x05\x01
33. Prefix string: "\x2A\x86\x48\x86\xF7\x14\x01\x05"
34. Prefix index: 10
35. ATTRTYP: 0x000a0001
36. OID: 0.9.2342.19200300.100.1.1 (uid attribute)
37. Binary: \x09\x92\x26\x89\x93\xF2\x2C\x64\x01\x01
38. Prefix string: "\x09\x92\x26\x89\x93\xF2\x2C\x64\x01"
39. Prefix index: 21
40. ATTRTYP: 0x00150001
41. OID: 2.16.840.1.113730.3.1.1 (carLicense attribute)
42. Binary: \x60\x86\x48\x01\x86\xF8\x42\x03\x01\x01
43. Prefix string: "\x60\x86\x48\x01\x86\xF8\x42\x03\x01"
44. Prefix index: 22
45. ATTRTYP: 0x00160001
46. OID: 1.2.840.113556.1.5.7000.53 (crossRefContainer class)
47. Binary: \x2A\x86\x48\x86\xF7\x14\x01\x05\xB6\x58\x35
48. Prefix string: "\x2A\x86\x48\x86\xF7\x14\x01\x05\xB6\x58"
49. Prefix index: 23
50. ATTRTYP: 0x00170035
51. OID: 2.5.21.2 (ditContentRules attribute)
52. Binary: \x55\x15\x02
53. Prefix string: "\x55\x15"
54. Prefix index: 24
55. ATTRTYP: 0x00180002
56. OID: 2.5.18.1 (createTimeStamp attribute)
57. Binary: \x55\x12\x01
58. Prefix string: "\x55\x12"
59. Prefix index: 25
60. ATTRTYP: 0x00190001
61. OID: 2.5.20.1 (subSchema class)
62. Binary: \x55\x14\x01
63. Prefix string: "\x55\x14"
64. Prefix index: 26
65. ATTRTYP: 0x001a0001

## ATTRVALBLOCK

The ATTRVALBLOCK structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a sequence of [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) values.

1. typedef struct {
2. [range(0,10485760)] ULONG valCount;
3. [size\_is(valCount)] ATTRVAL\* pAVal;
4. } ATTRVALBLOCK;

**valCount:**  The number of items in the pAVal array.

**pAVal:**  The sequence of attribute values.

## ATTRVALFromValue

1. procedure ATTRVALFromValue(v: Value,
2. s: Syntax,
3. var t: PrefixTable) : ATTRVAL

The ATTRVALFromValue procedure converts a value in the abstract [Value](#Section_c1b732d37bf94ba181ee07157f07294c) encoding *v* of syntax *s* into a concrete [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4), using the [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) represented by *t*. This procedure can mutate the supplied prefix table.

See section [5.16.3](#Section_0d7070d2f71647109f92812dc4cd8a53) for the specification of this procedure.

## BindToDSA()

1. procedure BindToDSA(dsa: DSNAME): DRS\_HANDLE

The BindToDSA procedure establishes an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) connection to the target [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) represented by its [**DSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8). It also performs the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) call. It returns the RPC handle on success or null on failure.

## BOOL

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a Boolean value, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.3.

## BYTE

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a single byte, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.6.

## CHANGE\_LOG\_ENTRIES

CHANGE\_LOG\_ENTRIES is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d), normatively specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.7.3; the type of the pmsgOut.V1.pLog field of the IDL\_DRSGetNT4ChangeLog response. The following five fields within this type are used in specifying IDL\_DRSGetNT4ChangeLog server behavior:

**Size**: MUST be 0x00000010.

**Version**: MUST be 0x00000001.

**SequenceNumber**: The sequence number for the buffer. MUST be set to 0x00000001 in a response to an IDL\_DRSGetNT4ChangeLog request with pmsgIn.V1.pRestart = null. The value of pmsgOut.V1.pRestart in any IDL\_DRSGetNT4ChangeLog response MUST encapsulate SequenceNumber. In a response to an IDL\_DRSGetNT4ChangeLog request with pmsgIn.V1.pRestart ≠ null, SequenceNumber is the value encapsulated in pmsgIn.V1.pRestart, plus one.

**Flags**: MUST be 0x00000000.

**ChangeLogEntries**: A pointer to an array of [CHANGELOG\_ENTRY](#Section_0861744b5ee0428fb11aa25092636b64).

## CHANGELOG\_ENTRY

CHANGELOG\_ENTRY is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) that is defined in [[MS-NRPC]](%5bMS-NRPC%5d.pdf#Section_ff8f970f3e3740f7bd4baf7336e4792f) section 3.5.4.6.4, with more information in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.7.1.2. The abstract variable [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).pdcChangeLog is a sequence of CHANGELOG\_ENTRY. The following two fields within this type are used in specifying IDL\_DRSGetNT4ChangeLog server behavior:

**ChangeLogEntrySize**: A [DWORD](#Section_60c3f5f194924d1083c89a155e162ef3) containing the size, in bytes, of the CHANGELOG\_ENTRY structure.

**SerialNumber**: A [LARGE\_INTEGER](#Section_ebf2c36755e84066b879f80f1e8f69a9) containing the serial number of the [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) represented in this CHANGELOG\_ENTRY.

## CheckGroupMembership

1. procedure CheckGroupMembership(
2. token: ClientAuthorizationInfo,
3. groupSid: SID): boolean

The CheckGroupMembership procedure returns true only if the user represented by *token* is a member of the [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) whose [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) is *groupSid*. For more details, see [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.5.3.

## ClientAuthorizationInfo

ClientAuthorizationInfo is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that represents a client's [**security context**](#gt_88d49f20-6c95-4b64-a52c-c3eca2fe5709) that contains authorization information for a client.

## ClientExtensions

1. procedure ClientExtensions(hDrs: DRS\_HANDLE): DRS\_EXTENSIONS\_INT

The ClientExtensions server procedure gets the client extensions presented in the IDL\_DRSBind call that created *hDrs*. Any fields not specified by the client in the *pextClient* parameter to IDL\_DRSBind (such that *pextClient^.cb* is less than the offset of the end of the field of [DRS\_EXTENSIONS\_INT](#Section_3ee529b123db4996948a042f04998e91)) are set to 0.

## ClientUUID

1. procedure ClientUUID(hDrs: DRS\_HANDLE): UUID

The ClientUUID procedure returns the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) that identifies the caller presented in the [IDL\_DRSBind](#Section_605b1ea19cdc428fab7a70120e020a3d) call that created *hDrs*.

## ConcretePTFromAbstractPT

1. procedure ConcretePTFromAbstractPT(
2. prefixTable: PrefixTable): SCHEMA\_PREFIX\_TABLE

*Informative summary of behavior*: The ConcretePTFromAbstractPT procedure translates abstract [PrefixTable](#Section_2789d96b50e8444d82d6523831556d76) to a [SCHEMA\_PREFIX\_TABLE](#Section_9b371267e8b84c69997902dae02e5e38) structure.

1. prefixCount: ULONG
2. concretePrefixTable: SCHEMA\_PREFIX\_TABLE
3. prefixCount := prefixTable.length
4. concretePrefixTable.PrefixCount := prefixCount
5. for i := 0 to (prefixTable.length - 1)
6. concretePrefixTable.pPrefixTableEntry[i].prefix :=
7. prefixTable[i].prefixString
8. concretePrefixTable.pPrefixTableEntry[i].ndx :=
9. prefixTable[i].prefixIndex
10. endfor
11. return concretePrefixTable

## ConfigNC

1. procedure ConfigNC(): DSName

The ConfigNC procedure returns the [**dsname**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a) of [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).configNC.

## dc, DC

A global variable that represents the state of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), as defined in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.1.9, and the type of that variable. That definition is repeated here for convenience:

type **DC** = [

*serverGuid*: [GUID](#Section_5e740f50e6a048c9bca800072e85d963),

*invocationId*: GUID,

*usn*: 64-bit integer,

*prefixTable*: [PrefixTable](#Section_2789d96b50e8444d82d6523831556d76),

*defaultNC*: [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210),

*configNC*: config NC replica,

*schemaNC*: [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093) NC replica,

*partialDomainNCs*: set of partial domain NC replica,

*appNCs*: set of application NC replica,

*pdcChangeLog*: [PdcChangeLog](#Section_24444ebf2e164050af0890c85e611234),

*nt4ReplicationState*: [NT4ReplicationState](#Section_4ee4212d910042aeab10c480bcce73a9),

*ldapConnections*: [LDAPConnections](#Section_3653bad8fb4d4edda52179b19c1c1e95),

*replicationQueue*: [ReplicationQueue](#Section_6226aaa1178d45ff9e17815556739595),

*kccFailedConnections*: [KCCFailedConnections](#Section_eaffa80d8baf4784898ee9fbc7bd8296),

*kccFailedLinks*: [KCCFailedLinks](#Section_fec285f37f034cfc89ac911f61c0c7d3),

*rpcClientContexts*: [RPCClientContexts](#Section_65d838f52f694c228b263340182dcde1),

*rpcOutgoingContexts*: [RPCOutgoingContexts](#Section_b9f465938a4041869ecebc2612b4c3f4),

*fLinkValueStampEnabled*: boolean,

*nt4EmulatorEnabled*: boolean,

*fEnableUpdates*: boolean,

*minimumGetChangesReplyVersion*: integer,

*minimumGetChangesRequestVersion*: integer

]

The *ldapConnections*, *replicationQueue*, *kccFailedConnections*, *kccFailedLinks*, *rpcClientContexts*, and *rpcOutgoingContexts* fields are volatile state. Each volatile field is set to the empty sequence on server startup. The other fields are persistent state, [**updated**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) by using transactions.

The variable dc is the only global variable in this specification. It contains the state of the server:

1. dc: DC

## DefaultNC

1. procedure DefaultNC(): DSName

The DefaultNC procedure returns the [**dsname**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a) of the [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).defaultNC.

## DelSubRef

1. procedure DelSubRef(childNC: DSName)

*Informative summary of behavior*: This procedure deletes a [**sub-ref object**](#gt_a4b4bece-8452-402c-99c6-12ebf0af0b58) for the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) *childNC*, if it exists.

1. parentNC: DSName
2. rt: ULONG
3. /\* If the sub-ref object is not instantiated, delete it \*/
4. if(IT\_UNINSTANT in childNC!instanceType)
5. then
6. rt:=RemoveObj(childNC, false)
7. /\* Ignore rt because there are no possible errors returned by RemoveObj
8. while deleting a subref object. RemoveObj always returns success in this
9. procedure \*/
10. else
11. /\* Otherwise, just prevent continuation referrals from being
12. \* generated by removing childNC from the parent's subRefs list.
13. \*/
14. parentNC := GetObjectNC(ChildNC)
15. parentNC!subRefs := parentNC!subRefs – {childNC}
16. endif

## DescendantObject

1. procedure DescendantObject(
2. ancestor: DSName, rdns: unicodestring): DSName

The DescendantObject procedure constructs a [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) string by concatenating *rdns* and *ancestor*.dn, and then verifies the existence of the descendant [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). It returns the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) if the descendant exists, and null otherwise.

## DomainNameFromDN

1. procedure DomainNameFromDN(
2. dn: unicodestring): unicodestring

The DomainNameFromDN procedure returns the [**fully qualified domain name (FQDN) (1)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) of the [**crossRef object**](#gt_353fac65-0774-4ba8-8081-eb4c963f94e7) identified by *dn*, or null if no matching crossRef object exists.

## DN

DN is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that is a *unicodestring* (section [3.4.3](#Section_fbe9988847824858b5f25b521a44d836)) that contains a [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the form specified in [[RFC2253]](https://go.microsoft.com/fwlink/?LinkId=90327).

## DNBinary

DNBinary is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that represents the [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) [SYNTAX\_DISTNAME\_BINARY](#Section_8eefc5ab6d2248b4bea163b53a81a3a9). It consists of the following tuple:

type DNBinary = [dn: [DSName](#Section_a0d5477a522946b9890a54b924d487d1), binary: sequence of [BYTE](#Section_545826e419454580961f0f0c0a47e797)]

## DomainNameFromNT4AccountName

1. procedure DomainNameFromNT4AccountName(
2. nt4AccountName: unicodestring): unicodestring

If *nt4AccountName* is a name in Windows NT 4.0 operating system account name format, that is, two components separated by a backslash (for example, "DOMAIN\username"), the DomainNameFromNT4AccountName procedure returns the first component (the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) name, or "DOMAIN" in this example). If the *nt4AccountName* is not in this format, null is returned.

## DRS\_EXTENSIONS

The DRS\_EXTENSIONS structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for capabilities information used in version negotiation.

1. typedef struct {
2. [range(1,10000)] DWORD cb;
3. [size\_is(cb)] BYTE rgb[];
4. } DRS\_EXTENSIONS;

**cb:**  The size, in bytes, of the **rgb** array.

**rgb:**  To [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331), this field is a string of **cb** bytes. It is interpreted by the client and the server as the first **cb** bytes of a [DRS\_EXTENSIONS\_INT](#Section_3ee529b123db4996948a042f04998e91) structure that follow the **cb** field of that structure. The fields of the DRS\_EXTENSIONS\_INT structure are in little-endian byte order. Since both DRS\_EXTENSIONS and DRS\_EXTENSIONS\_INT begin with a **DWORD** **cb**, a field in DRS\_EXTENSIONS\_INT is at the same offset in DRS\_EXTENSIONS as it is in DRS\_EXTENSIONS\_INT.

## DRS\_EXTENSIONS\_INT

The DRS\_EXTENSIONS\_INT structure is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for structured capabilities information used in version negotiation.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cb | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| dwFlags | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SiteObjGuid (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Pid | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| dwReplEpoch | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| dwFlagsExt | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ConfigObjGUID (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| dwExtCaps | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**cb (4 bytes):** The count of bytes in the fields **dwFlags** through **dwExtCaps**, inclusive.[<38>](#Appendix_A_38" \o "Product behavior note 38)[<39>](#Appendix_A_39" \o "Product behavior note 39)[<40>](#Appendix_A_40" \o "Product behavior note 40) This field allows the DRS\_EXTENSIONS\_INT structure to be extended by including new fields at the end of the structure.

**dwFlags (4 bytes):** The **dwFlags** field contains individual bit flags that describe the capabilities of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that produced the DRS\_EXTENSIONS\_INT structure.[<41>](#Appendix_A_41" \o "Product behavior note 41)

The following table lists the bit flags, which are presented in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| A E | U O | D C | D F | M V | R M | A S | B A S | S E | G R I | C B | I N R | D C 2 | L V R | A E 2 | K E | A N C | G C 6 | G M 2 | G C 5 | P B 3 | S H | T M | D C F | R 3 | R 2 | G C 10 | D F 2 | W B 3 | G R 6 | G R 5 | G C 8 |

**BAS (DRS\_EXT\_BASE, 0x00000001)**: Unused. SHOULD be 1 and MUST be ignored.

**AS (DRS\_EXT\_ASYNCREPL, 0x00000002)**: If present, signifies that the DC supports [DRS\_MSG\_REPADD\_V2](#Section_892d85776e1545ba8e7a022807ed8649).

**RM (DRS\_EXT\_REMOVEAPI, 0x00000004)**: If present, signifies that the DC supports [IDL\_DRSRemoveDsServer](#Section_d5c310ae347a49d4a78e6ffb2eecd581) and [IDL\_DRSRemoveDsDomain](#Section_aa3cfa46c737425aae65ecaf9efe7e84).

**MV (DRS\_EXT\_MOVEREQ\_V2, 0x00000008)**: If present, signifies that the DC supports [DRS\_MSG\_MOVEREQ\_V2](#Section_d77d56704a3549d2b6c8e314a7f76ff3).

**DF (DRS\_EXT\_GETCHG\_DEFLATE, 0x00000010)**: If present, signifies that the DC supports [DRS\_MSG\_GETCHGREPLY\_V2](#Section_677d8fab6aa143279b6f62a6ad7fcfa3).

**DC (DRS\_EXT\_DCINFO\_V1, 0x00000020)**: If present, signifies that the DC supports [IDL\_DRSDomainControllerInfo](#Section_668abdc81db741049deafeab05ff1736).

**UO (DRS\_EXT\_RESTORE\_USN\_OPTIMIZATION, 0x00000040)**: Unused. SHOULD be 1 and MUST be ignored.

**AE (DRS\_EXT\_ADDENTRY, 0x00000080)**: If present, signifies that the DC supports [IDL\_DRSAddEntry](#Section_06764fc54df64104b6afa92bdaa81f6e).

**KE (DRS\_EXT\_KCC\_EXECUTE, 0x00000100)**: If present, signifies that the DC supports [IDL\_DRSExecuteKCC](#Section_ad807917687b40d9abe2053af0246523).

**AE2 (DRS\_EXT\_ADDENTRY\_V2, 0x00000200)**: If present, signifies that the DC supports [DRS\_MSG\_ADDENTRYREQ\_V2](#Section_895157d524e24eaf9060148d67669e27).

**LVR (DRS\_EXT\_LINKED\_VALUE\_REPLICATION, 0x00000400)**: If present, signifies that the DC supports [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb), and this support is enabled.

**DC2 (DRS\_EXT\_DCINFO\_V2, 0x00000800)**: If present, signifies that the DC supports [DRS\_MSG\_DCINFOREPLY\_V2](#Section_f567e60501fe4228960e14647c29f668).

**INR (DRS\_EXT\_INSTANCE\_TYPE\_NOT\_REQ\_ON\_MOD, 0x00001000)**: Unused. SHOULD be 1 and MUST be ignored.

**CB (DRS\_EXT\_CRYPTO\_BIND, 0x00002000)**: A client-only flag. If present, it indicates that the [**security provider**](#gt_05fd3925-0672-4f24-9dd9-2b9d441eb333) used for the connection supports [**session keys**](#gt_4f67a585-fb00-4166-93e8-cf4abca8226d) through [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) (example, Kerberos connections with mutual [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) enable RPC to expose session keys, but NTLM connections do not enable RPC to expose session keys).

**GRI (DRS\_EXT\_GET\_REPL\_INFO, 0x00004000)**: If present, signifies that the DC supports [IDL\_DRSGetReplInfo](#Section_dd29f9ceb30b411ebd54b77634eded47).

**SE (DRS\_EXT\_STRONG\_ENCRYPTION, 0x00008000)**: If present, signifies that the DC supports additional 128-bit encryption for passwords over the wire. DCs MUST NOT replicate passwords to other DCs that do not support this extension.

**DCF (DRS\_EXT\_DCINFO\_VFFFFFFFF, 0x00010000)**: If present, signifies that the DC supports [DRS\_MSG\_DCINFOREPLY\_VFFFFFFFF](#Section_625c5133cb5b440a9f53232ae1b2dc3f).

**TM (DRS\_EXT\_TRANSITIVE\_MEMBERSHIP, 0x00020000)**: If present, signifies that the DC supports [IDL\_DRSGetMemberships](#Section_d5ace4527cdd4d50bb6439b4c55180a2).

**SH (DRS\_EXT\_ADD\_SID\_HISTORY, 0x00040000)**: If present, signifies that the DC supports [IDL\_DRSAddSidHistory](#Section_376230a5d8064ae5970af6243ee193c8).

**PB3 (DRS\_EXT\_POST\_BETA3, 0x00080000)**: Reserved. MUST be set to 1 and ignored.

**GC5 (DRS\_EXT\_GETCHGREQ\_V5, 0x00100000)**: If present, signifies that the DC supports [DRS\_MSG\_GETCHGREQ\_V5](#Section_fd24b73c7b8143af8c7765bc2e3181b7).

**GM2(DRS\_EXT\_GETMEMBERSHIPS2, 0x00200000)**: If present, signifies that the DC supports [IDL\_DRSGetMemberships2](#Section_d4e67cc32ee14b2b8055cebefc556252).

**GC6 (DRS\_EXT\_GETCHGREQ\_V6, 0x00400000)**: Unused. This bit was used for a pre-release version of Windows. No released version of Windows references it. This bit can be set or unset with no change in behavior.

**ANC (DRS\_EXT\_NONDOMAIN\_NCS, 0x00800000)**: If present, signifies that the DC supports [**application NCs**](#gt_53fee475-b07f-45e1-b4b7-c7ac0c1e7f6a).

**GC8 (DRS\_EXT\_GETCHGREQ\_V8, 0x01000000)**: If present, signifies that the DC supports [DRS\_MSG\_GETCHGREQ\_V8](#Section_4304bb4ae9b54c8a8731df4d6f9ab567).

**GR5 (DRS\_EXT\_GETCHGREPLY\_V5, 0x02000000)**: Unused. SHOULD be 1 and MUST be ignored.

**GR6 (DRS\_EXT\_GETCHGREPLY\_V6, 0x04000000)**: If present, signifies that the DC supports [DRS\_MSG\_GETCHGREPLY\_V6](#Section_1317a6545dd645ffaf73919cbc7fbb45).

**WB3 (DRS\_EXT\_WHISTLER\_BETA3, 0x08000000)**: If present, signifies that the DC supports [DRS\_MSG\_ADDENTRYREPLY\_V3](#Section_1eeb493e93f1424cb8ebca74e6f051a0), [DRS\_MSG\_REPVERIFYOBJ](#Section_b8cc0c8a711944068f3a4172dc512735), [DRS\_MSG\_GETCHGREPLY\_V7](#Section_26eaca610f1947e7b3042580e9870aa8), and [DRS\_MSG\_QUERYSITESREQ\_V1](#Section_A5E57FA8944144C6AF986437DB28D6D6).

**DF2 (DRS\_EXT\_W2K3\_DEFLATE, 0x10000000)**: If present, signifies that the DC supports the W2K3 AD deflation library.

**GC10 (DRS\_EXT\_GETCHGREQ\_V10, 0x20000000)**: If present, signifies that the DC supports [DRS\_MSG\_GETCHGREQ\_V10](#Section_92b1b77d205846e09e8c6664b96a0cf9).

**R2 (DRS\_EXT\_RESERVED\_FOR\_WIN2K\_OR\_DOTNET\_PART2, 0x40000000)**: Unused. MUST be 0 and ignored.

**R3 (DRS\_EXT\_RESERVED\_FOR\_WIN2K\_OR\_DOTNET\_PART3, 0x80000000)**: Unused. MUST be 0 and ignored.

**SiteObjGuid (16 bytes):** A [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1). The [**objectGUID**](#gt_ad613dff-e9c4-4cb6-ad6b-0ce52038ceb5) of the [**site object**](#gt_0ce6abc5-9823-4a69-bb30-12e42ff99629) of which the DC's [**DSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8) is a descendant. For non-DC client callers, this field SHOULD be set to zero.

**Pid (4 bytes):** A 32-bit, signed integer value that specifies the process identifier of the client. This is for informational and debugging purposes only. The assignment of this field is implementation-specific.[<42>](#Appendix_A_42" \o "Product behavior note 42)

**dwReplEpoch (4 bytes):** A 32-bit, unsigned integer value that specifies the [**replication epoch**](#gt_cb4c7bb2-7c28-4ce0-b5f6-de93a7e236d8). This value is set to zero by all client callers. The server sets this value by assigning the value of msDS-ReplicationEpoch from its nTDSDSA object. If **dwReplEpoch** is not included in DRS\_EXTENSIONS\_INT, the value is considered to be zero.[<43>](#Appendix_A_43" \o "Product behavior note 43)

**dwFlagsExt (4 bytes):** An extension of the **dwFlags** field that contains individual bit flags. These bit flags determine which extended capabilities are enabled in the DC that produced the DRS\_EXTENSIONS\_INT structure. For non-DC client callers, no bits SHOULD be set. If **dwFlagsExt** is not included in DRS\_EXTENSIONS\_INT, all bit flags are considered unset.

The following table lists the bit flags, which are presented in little-endian byte order.[<44>](#Appendix_A_44" \o "Product behavior note 44)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | X | R B | L H | D A | X | X | X | X | X | CID | X | G R 9 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**DA (DRS\_EXT\_ADAM, 0x00000001)**: If present, signifies that the DC supports [DRS\_MSG\_REPSYNC\_V1](#Section_d29519a5f85e4bd5907a0777ce0be29f), [DRS\_MSG\_UPDREFS\_V1](#Section_ee70be3308dc48b2a59943a50943c0e1), [DRS\_MSG\_INIT\_DEMOTIONREQ\_V1](#Section_d71cd7834d7b443cadfe6b3a77a19671), [DRS\_MSG\_REPLICA\_DEMOTIONREQ\_V1](#Section_8e459d5d129e40be8bc90872f763c61d), and [DRS\_MSG\_FINISH\_DEMOTIONREQ\_V1](#Section_687fcbf22e1f467a9a1ce2c34b3f7ce1).

**LH (DRS\_EXT\_LH\_BETA2, 0x00000002)**: If present, signifies that the DC supports the DRS\_SPECIAL\_SECRET\_PROCESSING and DRS\_GET\_ALL\_GROUP\_MEMBERSHIP flags as well as **InfoLevel** 3 in [DRS\_MSG\_DCINFOREQ\_V1](#Section_18b23122a1c24367a677592e0d4eef18).

**RB (DRS\_EXT\_RECYCLE\_BIN, 0x00000004)**: If present, signifies that the DC has enabled the [**Recycle Bin**](#gt_54624800-58f4-45e9-90bf-c9b52dcf98f3) [**optional feature**](#gt_785b66f1-22b3-450f-97aa-a24a39d04d47).

**GR9 (DRS\_EXT\_GETCHGREPLY\_V9, 0x00000100)**: If present, signifies that the DC supports [**DRS\_MSG\_GETCHGREPLY\_V9**](#Section_b9564a194500444ba99b0da1b08cdb6f).

**CID (DRS\_EXT\_RPC\_CORRELATIONID\_1, 0x00000400)**: If present, signifies that the DC supports DRS\_MSG\_GETCHGREQ\_V11 (section [4.1.10.2.8](#Section_cb2bab15950b48f8af00118e186a1311)), DRS\_MSG\_REPADD\_V3 (section [4.1.19.1.4](#Section_63f7638bc9514486ab3f3f086d9fb622)), DRS\_MSG\_REPSYNC\_V2 (section [4.1.23.1.3](#Section_f32ab3844894416793f0eda08703d76b)), and DRS\_MSG\_UPDREFS\_V2 (section [4.1.26.1.3](#Section_af174769feae42128e1ca6072b022e6c)).

**ConfigObjGUID (16 bytes):** A GUID. This field is set to zero by all client callers. The server sets this field by assigning it the value of the objectGUID of the [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625) [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). If **ConfigObjGUID** is not included in DRS\_EXTENSIONS\_INT, the value is considered to be the [**NULL GUID**](#gt_ba500a5b-8c29-467c-a335-0980c8b11304) value.[<45>](#Appendix_A_45" \o "Product behavior note 45)

**dwExtCaps (4 bytes):** A mask for the **dwFlagsExt** field that contains individual bit flags. These bit flags describe the potential extended capabilities of the DC that produced the DRS\_EXTENSIONS\_INT structure. For non-DC client callers, no bits SHOULD be set. If neither **dwFlagsExt** nor **dwExtCaps** is included in DRS\_EXTENSIONS\_INT, all bits in **dwExtCaps** are considered unset. If **dwFlagsExt** is included in DRS\_EXTENSIONS\_INT but **dwExtCaps** is not, all relevant bits in **dwExtCaps** (as explained below) are implicitly set.[<46>](#Appendix_A_46" \o "Product behavior note 46)

Each bit in **dwExtCaps** corresponds exactly to each bit in **dwFlagsExt**. If the DC that produced the DRS\_EXTENSIONS\_INT structure supports a capability described by a bit in the **dwFlagsExt** field (that is, the bit either is or could potentially be set), then the corresponding bit in **dwExtCaps** MUST be set. If a bit in **dwExtCaps** is not set, it is assumed that the corresponding bit in **dwFlagsExt** will not and cannot be set.

**Note**  The **dwExtCaps** field is relevant only for capabilities that are labeled as "optional features" in the bit descriptions of **dwFlagsExt**. The bits in **dwExtCaps** that correspond to capabilities in **dwFlagsExt** that are not labeled as "optional features" MUST NOT be different from the setting of the **dwFlagsExt** bits. Currently, the capabilities represented by the DA and LH bits fit into this category.

## DRS\_HANDLE

DRS\_HANDLE is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle (as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824)) for use in calls to methods in the drsuapi RPC interface.

This type is declared as follows:

1. typedef [context\_handle] void\* DRS\_HANDLE;

For the specification of IDL\_DRSBind, see section [4.1.3](#Section_605b1ea19cdc428fab7a70120e020a3d).

Methods in the dsaop RPC interface do not use context handles.

## DRS\_OPTIONS

DRS\_OPTIONS is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a set of options sent to and received from various drsuapi methods.

This type is declared as follows:

1. typedef unsigned long DRS\_OPTIONS;

It is a bit field, presented in little-endian byte order, that contains the following values.

Seven elements of the set are interpreted differently by different methods; such elements have multiple symbolic names.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| M R | P S | I S | W R | A L L / D R | A R | G C / U N | A S | F S / N S | S N / R F | N R R | G S / L O | G A | C O | T S | A S R / I E | I S N | S S | N S Y | R G / N D | S U | S Q | F S P | F S I | G P | S P | N N | U C | D P S | D A S | S F | P E |

**X**: Unused. MUST be zero and ignored.

**AS (DRS\_ASYNC\_OP, 0x00000001)**: Perform the operation asynchronously.

**GC (DRS\_GETCHG\_CHECK, 0x00000002)**: Treat ERROR\_DS\_DRA\_REF\_NOT\_FOUND and ERROR\_DS\_DRA\_REF\_ALREADY\_EXISTS as success for calls to [IDL\_DRSUpdateRefs (section 4.1.26)](#Section_a273bbcfaeca46088ad4127d3e597cd4).

**UN (DRS\_UPDATE\_NOTIFICATION, 0x00000002)**: Identifies a call to [IDL\_DRSReplicaSync](#Section_25c71d91051f4c26977fa70892f29b00) that was generated due to a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) notification. See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.5.1.6 for more details on replication notifications. This flag is ignored by the server.

**AR (DRS\_ADD\_REF, 0x00000004)**: Register a client [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) for notifications of [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) to the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

**ALL (DRS\_SYNC\_ALL, 0x00000008)**: Replicate from all server DCs.

**DR (DRS\_DEL\_REF, 0x00000008)**: Deregister a client DC from notifications of updates to the NC replica.

**WR (DRS\_WRIT\_REP, 0x00000010)**: Replicate a writable [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac), not a read-only partial replica or read-only full replica.

**IS (DRS\_INIT\_SYNC, 0x00000020):** Perform replication at startup.

**PS (DRS\_PER\_SYNC, 0x00000040):** Perform replication periodically.

**MR (DRS\_MAIL\_REP, 0x00000080)**: Perform replication using SMTP as a transport.

**ASR (DRS\_ASYNC\_REP, 0x00000100)**: Populate the NC replica asynchronously.

**IE (DRS\_IGNORE\_ERROR, 0x00000100)**: Ignore errors.

**TS (DRS\_TWOWAY\_SYNC, 0x00000200)**: Inform the server DC to replicate from the client DC.

**CO (DRS\_CRITICAL\_ONLY, 0x00000400)**: Replicate only [**system-critical objects**](#gt_97fbce29-609a-4a1e-83cf-e242ca23b21e).

**GA (DRS\_GET\_ANC, 0x00000800):** Include updates to [**ancestor objects**](#gt_874fa4dc-37f4-4467-91c3-78d5e4e5a410) before updates to their descendants.

**GS (DRS\_GET\_NC\_SIZE, 0x00001000)**: Get the approximate size of the server NC replica.

**LO (DRS\_LOCAL\_ONLY, 0x00001000)**: Perform the operation locally without contacting any other DC.

**NRR (DRS\_NONGC\_RO\_REP, 0x00002000)**: Replicate a read-only full replica. Not a writable or partial replica.

**SN (DRS\_SYNC\_BYNAME, 0x00004000)**: Choose the source server by network name.

**RF (DRS\_REF\_OK, 0x00004000)**: Allow the NC replica to be removed even if other DCs use this DC as a replication server DC.

**FS (DRS\_FULL\_SYNC\_NOW, 0x00008000)**: Replicate all updates in the [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16), even those that would normally be filtered.

**NS (DRS\_NO\_SOURCE, 0x00008000)**: The NC replica has no server DCs.

**FSI (DRS\_FULL\_SYNC\_IN\_PROGRESS, 0x00010000)**: When the flag DRS\_FULL\_SYNC\_NOW is received in a call to IDL\_DRSReplicaSync, the flag DRS\_FULL\_SYNC\_IN\_PROGRESS is sent in the associated calls to [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) until the replication cycle completes. This flag is ignored by the server.

**FSP (DRS\_FULL\_SYNC\_PACKET, 0x00020000)**: Replicate all updates in the replication request, even those that would normally be filtered.

**SQ (DRS\_SYNC\_REQUEUE, 0x00040000)**: This flag is specific to the Microsoft client implementation of IDL\_DRSGetNCChanges. It is used to identify whether the call was placed in the replicationQueue more than once due to implementation-specific errors. This flag is ignored by the server.

**SU (DRS\_SYNC\_URGENT, 0x00080000)**: Perform the requested replication immediately; do not wait for any timeouts or delays. For information about urgent replication, see [MS-ADTS] section 3.1.1.5.1.7.

**RG (DRS\_REF\_GCSPN, 0x00100000)**: Requests that the server add an entry to repsTo for the client on the root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of the NC replica that is being replicated. When repsTo is set using this flag, the notifying client DC contacts the server DC using the [**service principal name**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) that begins with "GC" (section [2.2.3.2](#Section_41efc56e00074e88bafed7af61efd91f)).

**ND (DRS\_NO\_DISCARD, 0x00100000)**: This flag is specific to the Microsoft implementation. It identifies when the client DC is to call the requested IDL\_DRSReplicaSync method individually, without overlapping other outstanding calls to IDL\_DRSReplicaSync. This flag is ignored by the server.

**NSY (DRS\_NEVER\_SYNCED, 0x00200000)**: There is no successfully completed replication from this source server.

**SS (DRS\_SPECIAL\_SECRET\_PROCESSING, 0x00400000):** Do not replicate [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) values of attributes that contain [**secret data**](#gt_0c8d49b7-bdf7-4824-a91f-481cb10c5052).

**ISN (DRS\_INIT\_SYNC\_NOW, 0x00800000):** Perform initial replication now.

**PE (DRS\_PREEMPTED, 0x01000000):** The replication attempt is preempted by a higher priority replication request.

**SF (DRS\_SYNC\_FORCED, 0x02000000):** Force replication, even if the replication system is otherwise disabled.

**DAS (DRS\_DISABLE\_AUTO\_SYNC, 0x04000000):** Disable replication induced by update notifications.

**DPS (DRS\_DISABLE\_PERIODIC\_SYNC, 0x08000000)**: Disable periodic replication.

**UC (DRS\_USE\_COMPRESSION, 0x10000000)**: Compress response messages.

**NN (DRS\_NEVER\_NOTIFY, 0x20000000)**: Do not send update notifications.

**SP (DRS\_SYNC\_PAS, 0x40000000):** Expand the partial attribute set of the partial replica.

**GP (DRS\_GET\_ALL\_GROUP\_MEMBERSHIP, 0x80000000)**: Replicate all kinds of [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) membership. If this flag is not present non[**universal group**](#gt_f46053d6-0708-4094-ac63-57c1bcb73d32) membership will not be replicated.

For information about the Windows versions in which these flags were introduced, supported, or deprecated, see the following behavior note.[<47>](#Appendix_A_47" \o "Product behavior note 47)

## DRS\_MORE\_GETCHGREQ\_OPTIONS

DRS\_MORE\_GETCHGREQ\_OPTIONS is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a set of extra options sent to the [IDL\_DRSGetNCChanges](#Section_b63730ac614c431c950128d6aca91894) method.

This type is declared as follows:

1. typedef unsigned long DRS\_MORE\_GETCHGREQ\_OPTIONS;

It is a bit field, presented in little-endian byte order, which contains the following values.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | X | X | X | T G T | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**TGT (DRS\_GET\_TGT, 0x00000001)**: Include [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) to the [**target object**](#gt_62c95f88-0024-410c-b008-b637d04803ad) of a [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) before updates to the link value.

## DRS\_SecBuffer

DRS\_SecBuffer is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a buffer that contains [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) data.

1. typedef struct {
2. [range(0,10000)] unsigned long cbBuffer;
3. unsigned long BufferType;
4. [size\_is(cbBuffer)] BYTE\* pvBuffer;
5. } DRS\_SecBuffer;

**cbBuffer:**  The size, in bytes, of the pvBuffer array.

**BufferType:**  A bit field, presented in little-endian byte order, that contains the following values:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | X | T Y P | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | R O | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**TYP**: Can be one of the following values:

| Value | Meaning |
| --- | --- |
| SECBUFFER\_EMPTY  0x00000000 | A placeholder in the buffer array. The caller can supply several such entries in the array, and the security package can return data in them. |
| SECBUFFER\_DATA  0x00000001 | Used for common data. The security package can read this data and write it, for example, to encrypt some or all of it. |
| SECBUFFER\_TOKEN  0x00000002 | This buffer is used to indicate the security token portion of the message. This is read-only for input parameters or read/write for output parameters. |
| SECBUFFER\_PKG\_PARAMS  0x00000003 | These are transport-to-package–specific parameters. For example, the Netware redirector can supply the [**server object**](#gt_62a8c543-5998-480b-8fa7-41a8f04a18e5) identifier, while DCE [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) can supply an association [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3), and so on. |
| SECBUFFER\_MISSING  0x00000004 | The security package uses this value to indicate the number of missing bytes in a particular message. The **pvBuffer** member is ignored in this type. |
| SECBUFFER\_EXTRA  0x00000005 | The security package uses this value to indicate the number of extra or unprocessed bytes in a message. |
| SECBUFFER\_STREAM\_TRAILER  0x00000006 | Indicates a protocol-specific trailer for a particular record. |
| SECBUFFER\_STREAM\_HEADER  0x00000007 | Indicates a protocol-specific header for a particular record. |

**RO (SECBUFFER\_READONLY, 0x80000000)**: The buffer is read-only. This flag is intended for sending header data to the security package for [**checksumming**](#gt_fa444149-ef93-4512-a278-2e756295630c). The package can read this buffer but cannot modify it.

**pvBuffer:**  Authentication data.

## DRS\_SecBufferDesc

**DRS\_SecBufferDesc** is a Generic Security Service (GSS) Kerberos [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) token, as specified in [[RFC1964]](https://go.microsoft.com/fwlink/?LinkId=90304).

1. typedef struct {
2. unsigned long ulVersion;
3. [range(0,10000)] unsigned long cBuffers;
4. [size\_is(cBuffers)] DRS\_SecBuffer\* Buffers;
5. } DRS\_SecBufferDesc;

**ulVersion:**  MUST be 0.

**cBuffers:**  The number of items in the Buffers array.

**Buffers:**  Buffers that contain authentication data.

## DRS\_SPN\_CLASS

A *unicodestring* constant (section [3.4.3](#Section_fbe9988847824858b5f25b521a44d836)) that is used as the [**service class**](#gt_c1386afd-9d42-47c7-a7ea-d01912a17647) in the [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) for a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). It has the value "E3514235-4B06-11D1-AB04-00C04FC2DCD2".

## DS\_REPL\_OP\_TYPE

DS\_REPL\_OP\_TYPE is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) operation type.

1. typedef enum
2. {
3. DS\_REPL\_OP\_TYPE\_SYNC = 0,
4. DS\_REPL\_OP\_TYPE\_ADD,
5. DS\_REPL\_OP\_TYPE\_DELETE,
6. DS\_REPL\_OP\_TYPE\_MODIFY,
7. DS\_REPL\_OP\_TYPE\_UPDATE\_REFS
8. } DS\_REPL\_OP\_TYPE;

**DS\_REPL\_OP\_TYPE\_SYNC:** Sync [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) from server [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**DS\_REPL\_OP\_TYPE\_ADD:** Add NC replica server DC.

**DS\_REPL\_OP\_TYPE\_DELETE:** Remove NC replica server DC.

**DS\_REPL\_OP\_TYPE\_MODIFY:** Modify NC replica server DC.

**DS\_REPL\_OP\_TYPE\_UPDATE\_REFS:** Update NC replica client DC.

## DSAObj

1. procedure DSAObj(): DSName

The DSAObj procedure returns the [**dsname**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a) of the [**DC's**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

1. return select one o from children ConfigNC()
2. where o!objectGUID = dc.serverGUID

## DSA\_RPC\_INST

The DSA\_RPC\_INST structure is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) that represents a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

1. typedef struct \_DSA\_RPC\_INST {
2. DWORD cb;
3. DWORD cbpszServerOffset;
4. DWORD cbpszAnnotationOffset;
5. DWORD cbpszInstanceOffset;
6. DWORD cbpguidInstanceOffset;
7. } DSA\_RPC\_INST,
8. \*PDSA\_RPC\_INST;

**cb:**  The total number of bytes in the DSA\_RPC\_INST structure.

**cbpszServerOffset:**  The offset from the start of the DSA\_RPC\_INST structure to a location that specifies the start of the server name of this instance.

**cbpszAnnotationOffset:**  The offset from the start of the DSA\_RPC\_INST structure to a location that specifies the start of the annotation of this instance.

**cbpszInstanceOffset:**  The offset from the start of the DSA\_RPC\_INST structure to a location that specifies the start of the NetworkAddress (section [5.134](#Section_3d0d3777935847ddb55534405f57f912)) of this instance.

**cbpguidInstanceOffset:**  The offset from the start of the DSA\_RPC\_INST structure to a location that specifies the start of the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) for the instance.

## DSName

DSName is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) for representing a [**dsname**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a). It corresponds to the concrete representation [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86). It consists of a tuple that identifies an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9). This tuple is discussed in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.1.5. For this document, the fields of the tuple are defined as follows:

1. type DSName = [dn: StringName , guid: GUID, sid: Sid]

The *dn* field corresponds to the **StringName** field of the DSNAME structure and contains the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the object.

The *guid* field corresponds to the **Guid** field of the DSNAME structure and contains the value of the object's objectGUID [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).

The *sid* field corresponds to the **Sid** field of the DSNAME structure. If the object possesses an objectSid attribute, it contains the value of the object's objectSid attribute. If the object does not possess an objectSid attribute, the field is null.

## DSNAME

DSNAME is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for representing a [DSName](#Section_a0d5477a522946b9890a54b924d487d1), identifying a [**directory object**](#gt_407dbc2c-3140-4e31-9085-0087e2d3bab2) using the values of one or more of its [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f): objectGUID, objectSid, or distinguishedName.

1. typedef struct {
2. unsigned long structLen;
3. unsigned long SidLen;
4. GUID Guid;
5. NT4SID Sid;
6. unsigned long NameLen;
7. [range(0, 10485761)] [size\_is(NameLen + 1)]
8. WCHAR StringName[];
9. } DSNAME;

**structLen:**  The length, in bytes, of the entire data structure.

**SidLen:**  The number of bytes in the Sid field used to represent the [**object's**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) objectSid attribute value. Zero indicates that the DSNAME does not identify the objectSid value of the directory object.

**Guid:**  The value of the object's objectGUID attribute specified as a GUID structure, which is defined in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.3.4. If the values for all fields in the GUID structure are zero, this indicates that the DSNAME does not identify the objectGUID value of the directory object.

**Sid:**  The value of the object's objectSid attribute, its [**security identifier**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d), specified as a **SID** structure, which is defined in [MS-DTYP] section 2.4.2. The size of this field is exactly 28 bytes, regardless of the value of **SidLen**, which specifies how many bytes in this field are used. Note that this is smaller than the theoretical size limit of a **SID**, which is 68 bytes. While Windows publishes a general SID format, Windows never uses that format in its full generality. 28 bytes is sufficient for a Windows SID.

**NameLen:**  The number of characters in the **StringName** field, not including the terminating null character, used to represent the object's distinguishedName attribute value. Zero indicates that the DSNAME does not identify the distinguishedName value of the directory object.

**StringName:**  A null-terminated [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) value of the object's distinguishedName attribute, as specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.1.4. This field always contains at least one character: the terminating null character. Each Unicode value is encoded as 2 bytes. The byte ordering is little-endian.[<48>](#Appendix_A_48" \o "Product behavior note 48)

The following table shows an alternative representation of this structure.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| structLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SidLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Guid.Data1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Guid.Data2 | | | | | | | | | | | | | | | | Guid.Data3 | | | | | | | | | | | | | | | |
| Guid.Data4... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ...Guid.Data4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Sid... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ...Sid... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ...Sid... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ...Sid... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ...Sid... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ...Sid... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ...Sid | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| NameLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| StringName (Variable Length) ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Note**  All fields have little-endian byte ordering.

### DSNAME Equality

When comparing [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) elements for equality, an implementation must be aware that multiple [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) can be specified. DSNAME values x and y are equal only if one of the following conditions holds:

* x.Guid is not zeros and y.Guid is not zeros and x.Guid = y.Guid
* All of the following are true:
  + x.Guid is zeros or y.Guid is zeros.
  + x.StringLen ≠ 0.
  + The number of [**RDNs**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) in x is the same as in y.
  + For each RDN xi in x and RDN yi in y (see [[RFC2253]](https://go.microsoft.com/fwlink/?LinkId=90327)):
    - AttributeType of xi = AttributeType of yi.
    - AttributeValue of xi = AttributeValue of yi, without regard to case differences, Hiragana and Katakana character differences, and nonspacing characters.
* All of the following are true:
  + x.Guid is zeros.
  + y.Guid is zeros.
  + x.StringLen = 0.
  + y.StringLen = 0.
  + x.SidLen ≠ 0.
  + x.SidLen = y.SidLen.
  + x.Sid and y.Sid contain identical values in the first x.SidLen array items.

## DSTIME

**DSTIME** is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for time expressed as the number of seconds since January 1, 1601, 12:00:00 A.M.

This type is declared as follows:

1. typedef LONGLONG DSTIME;

The following diagram shows an alternative representation of this type.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| cSeconds... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ...cSeconds | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Note**  Byte ordering is little-endian.

## DWORD

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a 32-bit, unsigned integer, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.9.

## ENTINF

ENTINF is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the identity and [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) (some or all) of a given [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

1. typedef struct {
2. DSNAME\* pName;
3. unsigned long ulFlags;
4. ATTRBLOCK AttrBlock;
5. } ENTINF;

**pName:**  The identity of the object.

**ulFlags:**  A flags field that supports the following flags, which are presented in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | X | X | D O | M | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | R M | X | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**M (ENTINF\_FROM\_MASTER, 0x00000001)**: Retrieved from a full [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac).

**DO (ENTINF\_DYNAMIC\_OBJECT, 0x00000002)**: A [**dynamic object**](#gt_ea6b6f3f-6bed-4622-aaca-fd7df28badb9).

**RM (ENTINF\_REMOTE\_MODIFY, 0x00010000)**: A remote modify request to IDL\_DRSAddEntry (section [4.1.1.3](#Section_ec6234f623dd4d9bac24030093a3e039)).

**AttrBlock:**  Some of all of the attributes for this object, as determined by the particular method. See section [1.3.3](#Section_b8c1a431335c4797a5d294569dec581b) for an overview of methods using type ENTINF.

## ENTINF\_GetValue

1. procedure ENTINF\_GetValue (
2. entInf: ENTINF,
3. attribute: ATTRTYP,
4. prefixTable: PrefixTable): ATTRVAL

*Informative summary of behavior*: The ENTINF\_GetValue procedure scans an [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b) structure and returns the first [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) structure for the requested *attribute*. The *attribute* parameter is based on [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).prefixTable, while the [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) within *entInf* are based on the *prefixTable* parameter.

1. attrType: ATTRTYP
2. oid : OID
3. oid := OidFromAttid(dc.prefixTable, attribute)
4. attrType := MakeAttid(prefixTable, oid)
5. for each i in [0 .. entInf.AttrBlock.attrCount -1] do
6. if (entInf.AttrBlock.pAttr[i].attrTyp = attrType) and
7. (entInf.AttrBlock.pAttr[i].AttrVal.valCount > 0) then
8. return entInf.AttrBlock.pAttr[i].AttrVal.pAVal[0]
9. endif
10. endfor
11. return null

## ENTINF\_SetValue

1. procedure ENTINF\_SetValue (
2. var entInf: ENTINF,
3. attribute: ATTRTYP,
4. attrVal: ATTRVAL,
5. prefixTable: PrefixTable)

The ENTINF\_SetValue procedure [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) value within the [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b). If *attrVal* is null, then the attribute is removed from the list (if it exists). If the value is non-null, then the attribute value is updated or added to the list (when a value is not already present). The *attribute* and *attrVal* parameters are based on [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).prefixTable, while the attributes within *entInf* are based on the *prefixTable* parameter.

## ENTINF\_EnumerateAttributes

1. procedure ENTINF\_EnumerateAttributes(
2. e: ENTINF,
3. prefixTable: PrefixTable): set of ATTRTYP

The ENTINF\_EnumerateAttributes procedure returns the list of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) (based on [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).prefixTable) that are present in the [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b) *e*. Attributes within *e* are based on the *prefixTable* parameter.

## ENTINFLIST

ENTINFLIST is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a list of [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b) entries.

1. typedef struct ENTINFLIST {
2. struct ENTINFLIST\* pNextEntInf;
3. ENTINF Entinf;
4. } ENTINFLIST;

**pNextEntInf:**  The next ENTINFLIST in the sequence, or null.

**Entinf:**  An ENTINF entry.

## Expunge

1. procedure Expunge(obj: DSName)

The Expunge procedure physically removes an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) whose [DSName](#Section_a0d5477a522946b9890a54b924d487d1) is *obj* from the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9), without enforcing referential integrity constraints. The object is immediately removed without undergoing conversion to a [**tombstone**](#gt_9d8e0963-13fa-4e19-a97f-7ce6bc90d20f).

## FILETIME

FILETIME is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a time, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.3.3.

## FilteredGCPAS

1. procedure FilteredGCPAS() : PARTIAL\_ATTR\_VECTOR\_V1\_EXT^

*Informative summary of behavior*: The FilteredGCPAS procedure returns a reference to an instance of structure [PARTIAL\_ATTR\_VECTOR\_V1\_EXT](#Section_1d5c1b34daa44761a8b5d3c0146a0e30) that contains the list of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that can be present, based on the [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093), on a filtered [**GC**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169) [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac).

1. attrSetSeq: sequence of DSName
2. filteredAttributeSet: sequence of ATTRTYP
3. pPartialAttrVector: PARTIAL\_ATTR\_VECTOR\_V1\_EXT^
4. attrId: ATTRTYP
5. i, j:int
6. attrSetSeq := select o from subtree SchemaNC() where
7. (attributeSchema in o!objectClass) and
8. (o!isMemberOfPartialAttributeSet = true)
9. filteredAttributeSet := GetFilteredAttributeSet()
10. pPartialAttrVector = new PARTIAL\_ATTR\_VECTOR\_V1\_EXT sized to hold
11. (attrSetSeq.length - filteredAttributeSet.length) entries in
12. its rgPartialAttr field
13. pPartialAttrVector^.dwVersion := 1
14. -
15. j := 0
16. for i := 0 to attrSetSeq.length-1
17. attrId = AttrtypFromSchemaObj(attrSetSeq[i]);
18. if (not attrId in filteredAttributeSet) then
19. /\* attribute is not in the filtered list \*/
20. partialAttrVector^.rgPartialAttr[j]:= attrId
21. j := j + 1
22. endif
23. endfor
24. pPartialAttrVector^.cattrs := j
25. return pPartialAttrVector^

## FilteredPAS

1. procedure FilteredPAS() : PARTIAL\_ATTR\_VECTOR\_V1\_EXT

*Informative summary of behavior*: The FilteredPAS procedure returns a reference to an instance of structure [PARTIAL\_ATTR\_VECTOR\_V1\_EXT](#Section_1d5c1b34daa44761a8b5d3c0146a0e30) that contains the list of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that can be present, based on the [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093), on a filtered [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

1. attrSetSeq: sequence of DSName
2. filteredAttributeSet: sequence of ATTRTYP
3. pPartialAttrVector: PARTIAL\_ATTR\_VECTOR\_V1\_EXT^
4. attrId: ATTRTYP
5. i, j: int
6. attrSetSeq := select o from subtree SchemaNC() where
7. (attributeSchema in o!objectClass) and
8. (o!systemFlags ∩
9. {FLAG\_ATTR\_NOT\_REPLICATED,
10. FLAG\_ATTR\_IS\_CONSTRUCTED} = null)
11. filteredAttributeSet := GetFilteredAttributeSet()
12. pPartialAttrVector = new PARTIAL\_ATTR\_VECTOR\_V1\_EXT sized to hold
13. (attrSetSeq.length - filteredAttributeSet.length) entries in
14. its rgPartialAttr field
15. pPartialAttrVector^.dwVersion := 1
16. for i := 0 to attrSetSeq.length-1
17. attrId = AttrtypFromSchemaObj(attrSetSeq [i]);
18. if (not attrId in filteredAttributeSet = null) then
19. /\* attribute is not in the filtered list \*/
20. pPartialAttrVector^.rgPartialAttr[j]:= attrId
21. j := j + 1
22. endif
23. endfor
24. pPartialAttrVector^.cAttrs := j
25. return pPartialAttrVector^

## FindChar

1. procedure FindChar(
2. s: unicodestring, start: integer, c: UCHAR): integer

*Informative summary of behavior*: The FindChar procedure returns the zero-based index of the first occurrence of *c* in the portion of *s* between the *start* and the end of *s*.

If *s* = null, *start* < 0 or *start* > *s*.length-1, this procedure returns -1. Otherwise, let *s* be represented as the sequence of characters {*s*[0], ... *s*[*s*.length - 1]}. Let i be such that i >= *start*, i <= *s*.length -1, *s*[i] = *c*, and *s*[start] ≠ *c*, ..., *s*[i-1] ≠ *c*. If such an *i* exists, this procedure returns *i*. Otherwise, this procedure returns -1.

## FindCharRev

1. procedure FindCharRev(
2. s: unicodestring,
3. start: integer,
4. c: UCHAR): integer

*Informative summary of behavior*: The FindCharRev procedure returns the zero-based index of the last occurrence of *c* in the portion of s between the start and the end of *s*.

If *s* = null, *start* < 0 or *start* > *s*.length-1, this procedure returns -1. Otherwise, let *s* be represented as the sequence of characters {*s*[0], ... *s*[*s*.length - 1]}. Let i be such that *i* >= *start*, *i* <= *s*.length -1, *s*[i] = *c*, and s[i+1] ≠ *c*, ..., *s*[*s*.length - 1] ≠ *c*. If such an *i* exists, this procedure returns *i*. Otherwise, this procedure returns -1.

## FOREST\_TRUST\_INFORMATION

FOREST\_TRUST\_INFORMATION is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for state information about trust relationships with other [**forests**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). This data is stored in [**objects of class**](#gt_c2c67596-8d8f-42b9-9c70-1c4f7c952200) trustedDomain in the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) of the forest root domain. Specifically, the msDS-TrustForestTrustInfo [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on such [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) contains information about the trusted forest or realm. The structure of the information contained in this attribute is represented in the following manner.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| Version | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| RecordCount | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Records (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Version (4 bytes):**  The version of the data structure. The only supported version of the data structure is 1.

**RecordCount (4 bytes):** The number of records present in the data structure.

**Records (variable):** Variable-length records that each contain a specific type of data about the forest trust relationship.

**Note**  Records are not necessarily aligned to 32-bit boundaries. Each record starts at the next byte after the previous record ends.

Each record is represented as described in section [5.64.1](#Section_2a16b808322f433ab5a671eefba82b5a).

**Note**  All fields have little-endian byte ordering.

### Record

Each Record is represented in the following manner.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| RecordLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Flags | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Timestamp | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| RecordType | | | | | | | | ForestTrustData (variable) | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**RecordLen (4 bytes):** The length, in bytes, of the entire record.

**Flags (4 bytes):** Individual bit flags that control how the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) trust information in this record can be used.

If **RecordType** = 0 or 1, the **Flags** field can have one or more of the following bits, which are presented in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | X | T D C | T D A | T DN | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**X**: Unused. Must be zero and ignored.

**TDN (LSA\_TLN\_DISABLED\_NEW, 0x00000001)**: The entry is not yet enabled.

**TDA (LSA\_TLN\_DISABLED\_ADMIN, 0x00000002)**: The entry is disabled by the administrator.

**TDC (LSA\_TLN\_DISABLED\_CONFLICT, 0x00000004)**: The entry is disabled due to a conflict with another trusted [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

If **RecordType** = 2, the **Flags** field can have one or more of the following bits, which are presented in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | N D C | N D A | S D C | S D A | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**SDA (LSA\_SID\_DISABLED\_ADMIN, 0x00000001)**: The entry is disabled for [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d)-based matches by the administrator.

**SDC (LSA\_SID\_DISABLED\_CONFLICT, 0x00000002)**: The entry is disabled due to a SID conflict with another trusted domain.

**NDA (LSA\_NB\_DISABLED\_ADMIN, 0x00000004)**: The entry is disabled for NetBIOS name-based matches by the administrator.

**NDC (LSA\_NB\_DISABLED\_CONFLICT, 0x00000008)**: The entry is disabled due to a NetBIOS domain name conflict with another trusted domain.

For RecordType = 2, NETBIOS\_DISABLED\_MASK is defined as a mask on the lower 4 bits of the **Flags** field.

For all record types, LSA\_FTRECORD\_DISABLED\_REASONS is defined as a mask on the lower 16 bits of the **Flags** field. Unused bits covered by the mask are reserved for future use.

**Timestamp (8 bytes):** A [FILETIME (section 5.59)](#Section_70ee934bc9b944498aa36dfe9cef3eff) that contains the time when this entry was created.

**RecordType (1 byte):** An 8-bit value that specifies the type of record contained in this specific entry. The allowed values are specified in section [5.65](#Section_5514b72b8452446aaa64abb35536baca).

**ForestTrustData (variable):** A variable length, type-specific record, depending on the RecordType value, that contains the specific type of data about the forest trust relationship.

**Important**  The type-specific ForestTrustData record is not necessarily aligned to a 32-bit boundary. Each record starts at the byte following the **RecordType** field.

There are three different type-specific records. Depending on the value of the **RecordType** field, the structure of the type-specific record differs as described below.

* If RecordType = 0 or RecordType = 1, then the type-specific record is represented in the following manner.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| NameLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Name (variable length)... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**NameLen**: The length, in bytes, of the **Name** field.

**Name**: The top-level name of the trusted forest, in UTF-8 format.

* If RecordType = 2, then the type-specific record is represented in the following manner. Note that the record contains the following structures one after another. It is important to note that none of the data shown below is necessarily aligned to 32-bit boundaries.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| SidLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Sid (variable length)... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DnsNameLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DnsName (variable length)... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| NetbiosNameLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| NetbiosName (variable length)... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**SidLen**: The length, in bytes, of the **Sid** field.

**Sid**: The SID of a domain in the trusted forest, specified as a [SID](#Section_13560cc227ff43a09d6fd686bccc5f3c) structure, which is defined in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2.

**DnsNameLen**: The length, in bytes, of the **DnsName** field.

**DnsName**: The [**FQDN (1)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) of a domain in the trusted forest, in UTF-8 format.

**NetbiosNameLen**: The length, in bytes, of the **NetbiosName** field.

**NetbiosName**: The NetBIOS name of a domain in the trusted forest, in UTF-8 format.

* If RecordType is not one of the preceding values, then the type-specific record is represented in the following manner.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| BinaryDataLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BinaryData (variable length)... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**BinaryDataLen**: The length, in bytes, of the **BinaryData** field.

**BinaryData**: Trusted forest data.

### Determining If a Name Is in a Trusted Forest

This section describes procedures that use the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) trust information contained in the msDS-TrustForestTrustInfo [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) to determine if a given [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) is in a trusted forest.

The procedures described in this section use the following data structures.

1. struct {
2. ULONG RecordCount;
3. PX\_FOREST\_TRUST\_RECORD \*Entries;
4. } X\_FOREST\_TRUST\_INFORMATION;
6. struct {
7. ULONG Flags;
8. FOREST\_TRUST\_RECORD\_TYPE ForestTrustType;
9. LARGE\_INTEGER Time;
10. union {
11. LPWSTR TopLevelName;
12. X\_FOREST\_TRUST\_DOMAIN\_INFO DomainInfo;
13. X\_FOREST\_TRUST\_BINARY\_DATA Data;
14. } ForestTrustData;
15. } X\_FOREST\_TRUST\_RECORD, \*PX\_FOREST\_TRUST\_RECORD;
16. struct {
17. SID \*Sid;
18. LPWSTR DnsName;
19. LPWSTR NetbiosName;
20. } X\_FOREST\_TRUST\_DOMAIN\_INFO;
21. struct {
22. ULONG Length;
23. BYTE \*Buffer;
24. } X\_FOREST\_TRUST\_BINARY\_DATA;

The X\_FOREST\_TRUST\_INFORMATION structure previously defined is used by the procedure to determine if a given domain is in a trusted forest. To unmarshal the content of the msDS-TrustForestTrustInfo attribute into this structure, the UnmarshalForestTrustInfo procedure described below can be used.

1. procedure ExtractString(
2. buffer: sequence of BYTE,
3. index: DWORD, size: DWORD): unicodestring;

The sequence [*index* .. *index* + *size*] of bytes in *buffer* is interpreted as a UTF-8 string, and a corresponding *unicodestring* (section [3.4.3](#Section_fbe9988847824858b5f25b521a44d836)) is returned.

1. procedure ExtractSid(
2. buffer: sequence of BYTE,
3. index: DWORD, size: DWORD): SID;

The sequence [*index* .. *index* + *size*] of bytes in *buffer* is converted into a [SID](#Section_13560cc227ff43a09d6fd686bccc5f3c) structure and returned.

1. procedure ExtractBinary(
2. buffer: sequence of BYTE,
3. index: DWORD, size: DWORD): sequence of BYTE;

The sequence [*index* .. *index* + *size*] of bytes in *buffer* is returned.

1. procedure UnmarshalForestTrustInfo
2. (inputBuffer: sequence of BYTE,
3. var forestTrustInfo: X\_FOREST\_TRUST\_INFORMATION): boolean

*Informative summary of behavior*: The UnmarshalForestTrustInfo procedure unmarshals the byte stream *inputBuffer*, which holds the content of a msDS-TrustForestTrustInfo attribute that contains forest trust information, as described in FOREST\_TRUST\_INFORMATION, into the forestTrustInfo structure.

1. index: DWORD
2. pdwVersion: ADDRESS OF DWORD
3. pdwRecordCount: ADDRESS OF DWORD
4. i: DWORD
5. pwdRecordLength: ADDRESS OF DWORD
6. pTrustRecord: ADDRESS OF X\_FOREST\_TRUST\_RECORD
7. pulTime: ADDRESS OF ULONGLONG
8. pType: ADDRESS OF BYTE
9. pSid: ADDRESS OF SID
10. pString: ADDRESS OF unicodestring
11. pdwSize: ADDRESS OF DWORD
12. index := 0
13. pdwVersion := ADR(inputBuffer[index])
14. if pdwVersion^ ≠ 1 then
15. return false
16. endif
18. index := index + 4
19. pdwRecordCount := ADR(inputBuffer[index])
20. forestTrustInfo.RecordCount := pdwRecordCount^
21. index := index + 4
22. /\* Extract each record \*/
23. for i:= 0 to pdwRecordCount^
24. /\* First 4 bytes of the record is the length \*/
25. pdwRecordLength := ADR(inputBuffer[index])
26. index := index + 4
27. pTrustRecord := forestTrustInfo.Entries[i]
28. /\* Next 4 bytes of the record are the flags \*/
29. pdwFlags := ADR(inputBuffer[index])
30. pTrustRecord^.Flags := pdwFlags^
31. index := index + 4
32. /\* Next 8 bytes of the record represent the Time field \*/
33. pulTime := ADR(inputBuffer[index])
34. pTrustRecord^.Time := pulTime^
35. index := index + 8
36. /\* Next byte represents trust type \*/
37. pType := ADR(inputBuffer[index])
38. pTrustRecord^.ForestTrustType := pType^
39. index := index + 1
40. if (pTrustRecord^.ForestTrustType = ForestTrustTopLevelName or
41. pTrustRecord^.ForestTrustType = ForestTrustTopLevelNameEx)
42. then
43. /\* Next 4 bytes represent the size of the top level name \*/
44. pdwSize := ADR(inputBuffer[index])
45. index := index + 4
46. /\* Extract the top level name; index is at the start of name \*/
47. pTrustRecord^.TopLevelName :=
48. ExtractString(inputBuffer, index, pdwSize^)
49. index := index + pdwSize^
50. else
51. if (pTrustRecord^.ForestTrustType = ForestTrustDomainInfo)
52. then
53. /\* Next 4 bytes represent the size of the sid \*/
54. pdwSize := ADR(inputBuffer[index])
55. index := index + 4
56. /\* Extract the sid; index is at the start of sid \*/
57. pTrustRecord^.DomainInfo.Sid :=
58. ExtractSid(inputBuffer, index, pdwSize^)
59. index := index + pdwSize^
60. /\* Next 4 bytes represent the size of the dns domain name \*/
61. pdwSize := ADR(inputBuffer[index])
62. index := index + 4
63. /\* Extract the dns domain name; index is at start of name \*/
64. pTrustRecord^.DomainInfo.DnsName :=
65. ExtractString(inputBuffer, index, pdwSize^)
66. index := index + pdwSize^
67. /\* Next 4 bytes represent the size of the netbios
68. \* domain name \*/
69. pdwSize := ADR(inputBuffer[index])
70. index := index + 4
71. /\* Extract the netbios domain name; index is at the start
72. \* of name \*/
73. pTrustRecord^.DomainInfo.NetbiosName :=
74. ExtractString(inputBuffer, index, pdwSize^)
75. index := index + pdwSize^
76. else
77. /\* Next 4 bytes represent the size of the binary data \*/
78. pdwSize := ADR(inputBuffer[index])
79. pTrustRecord^.Data.Length := pdwSize^
80. index := index + 4
81. /\* Extract the binary data; index is at the start of data \*/
82. pTrustRecord^.Data.Buffer :=
83. ExtractBinary(inputbuffer, index, pdwSize^)
84. index := index + pdwSize^
85. endif
87. endif
88. /\* index is now at the beginning of the next record \*/
89. endfor
90. return true

The following procedures are used to determine if a given domain *name*, [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d), or UPN is in a trusted forest. Since they make use of forest trust information data stored in [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) of the forest root domain (see FOREST\_TRUST\_INFORMATION), these functions only work on [**GC servers**](#gt_a5a99ce4-e206-42dc-8874-e103934c5b0d) or [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) in the forest root domain.

1. procedure IsDomainNameInTrustedForest(name: unicodestring,
2. referredDomain: unicodestring): boolean

*Informative summary of behavior*: The IsDomainNameInTrustedForest procedure determines if the domain with the name given by *name* is in a trusted forest. The input *name* can be a DNS or a NetBIOS name.

1. if IsDomainDnsNameInTrustedForest(name, referredDomain) then
2. return true
3. endIf
4. if IsDomainNetbiosNameInTrustedForest(name, referredDomain) then
5. return true
6. endIf
7. return false
8. procedure IsDomainSidInTrustedForest(sid: SID): boolean

*Informative summary of behavior*: The IsDomainSidInTrustedForest procedure determines if the domain with the SID given by *sid* is in a trusted forest.

1. tdos: set of DSName
2. f: X\_FOREST\_TRUST\_INFORMATION
3. b: boolean
4. tdos := select all o in Children ForestRootDomainNC() where
5. trustedDomain in o!objectClass and
6. o!trustAttributes & 0x00000008 ≠ 0 and
7. o!msDS-TrustForestTrustInfo ≠ null
8. foreach o in tdos
9. if not UnmarshalForestTrustInfo(o!msDS-TrustForestTrustInfo, f)
10. then
11. return false
12. else
13. foreach e in f.Entries
14. if (e.ForestTrustType = ForestTrustDomainInfo and
15. e.DomainInfo.Sid = sid and
16. LSA\_FTRECORD\_DISABLED\_REASONS not in e.Flags) then
17. b := true
18. foreach g in f.Entries
19. if (g.ForestTrustType = ForestTrustTopLevelNameEx and
20. LSA\_FTRECORD\_DISABLED\_REASONS not in g.Flags and
21. (g.TopLevelName = e.DomainInfo.DnsName or
22. IsSubdomainOf(e.DomainInfo.DnsName, g.TopLevelName))) then
23. b := false
24. break
25. endif
26. endfor
27. if b then
28. return true
29. endif
30. endif
31. endfor
32. endif
33. endfor
34. return false
35. procedure IsUPNInTrustedForest(upn: unicodestring): boolean

*Informative summary of behavior*: The IsUPNInTrustedForest procedure determines if the domain containing the account with the UPN given by *upn* is in a trusted forest.

1. interpret upn as being in the format "username@domainName"
2. return IsNamespaceInTrustedDomain(domainName, trustedForestName)

The IsDomainNameInTrustedForest procedure uses the following helper procedures to determine if a domain is in a trusted forest.

1. procedure IsSubdomainOf(subdomainName: unicodestring,
2. superiordomainName: unicodestring): boolean

The IsSubdomainOf procedure takes a pair of [**domain names**](#gt_45a1c9f1-0263-49a8-97c7-7aca1a99308c) and returns true if *subdomainName* is a subdomain of *superiordomainName* as described in [[RFC1034]](https://go.microsoft.com/fwlink/?LinkId=90263) section 3.1, and false otherwise.

1. procedure ForestTrustOwnsName(f: X\_FOREST\_TRUST\_INFORMATION, name: unicodestring): boolean
2. /\* if the name matches or is a subdomain of one in the exclusion list, the
3. \* forest does not own this name \*/
4. foreach e in f.Entries
5. if (e.ForestTrustType = ForestTrustTopLevelNameEx and
6. (e.TopLevelName = name or
7. IsSubdomainOf(name, e.TopLevelName))) then
8. return false
9. endif
10. endfor
12. /\* if a suffix of the name is in the inclusion list and is
13. \* not disabled, the forest owns this name \*/
14. foreach e in f.Entries
15. if (e.ForestTrustType = ForestTrustTopLevelName and
16. LSA\_FTRECORD\_DISABLED\_REASONS not in e.Flags and
17. (e.TopLevelName = name or
18. IsSubdomainOf(name, e.TopLevelName))) then
19. return true
20. endif
21. endfor
22. return false
23. procedure IsDomainDnsNameInTrustedForest(name: unicodestring,
24. var referredDomain: unicodestring) : boolean
25. tdos: set of DSName
26. f: X\_FOREST\_TRUST\_INFORMATION
27. /\* Get all the objects that represent trusted domains \*/
28. tdos := select all o in Children ForestRootDomainNC() where
29. trustedDomain in o!objectClass and
30. o!trustAttributes & 0x00000008 ≠ 0 and
31. o!msDS-TrustForestTrustInfo ≠ null
32. foreach o in tdos
33. if not UnmarshalForestTrustInfo(o!msDS-TrustForestTrustInfo, f)
34. then
35. return false
36. else
37. foreach e in f.Entries
38. if (e.ForestTrustType = ForestTrustDomainInfo and
39. e.DomainInfo.DnsName = name and
40. LSA\_SID\_DISABLED\_ADMIN not in e.Flags and
41. LSA\_SID\_DISABLED\_CONFLICT not in e.Flags and
42. ForestTrustOwnsName(f, e.DomainInfo.DnsName) then
43. referredDomain := o!trustPartner
44. return true
45. endif
46. endfor
47. endif
48. endfor
49. return false
50. procedure IsDomainNetbiosNameInTrustedForest
51. (name: unicodestring, var referredDomain: unicodestring): boolean
52. tdos: set of DSName
53. f: X\_FOREST\_TRUST\_INFORMATION
54. /\* Get all the objects that represent trusted domains \*/
55. tdos := select all o in Children ForestRootDomainNC() where
56. trustedDomain in o!objectClass and
57. o!trustAttributes & 0x00000008 ≠ 0 and
58. o!msDS-TrustForestTrustInfo ≠ null
59. foreach o in tdos
60. if not UnmarshalForestTrustInfo(o!msDS-TrustForestTrustInfo, f)
61. then
62. return false
63. else
64. foreach e in f.Entries
65. if (e.ForestTrustType = ForestTrustDomainInfo and
66. e.DomainInfo.NetbiosName = name and
67. NETBIOS\_DISABLED\_MASK not in e.Flags and
68. ForestTrustOwnsName(f, e.DomainInfo.DnsName) then
69. referredDomain := o!trustPartner
70. return true
71. endif
72. endfor
73. endif
74. endfor
75. return false

The IsUPNInTrustedForest procedure uses the following helper procedure to determine if a UPN is in a trusted forest.

1. procedure IsNamespaceInTrustedDomain
2. (name: unicodestring, var trustedForestName: unicodestring): boolean
3. tdos: set of DSName
4. f: X\_FOREST\_TRUST\_INFORMATION
5. b: boolean
6. dnsParent: unicodestring
7. parents: set of unicodestring
8. /\* if name is A.B.C, parents has the values A.B.C, B.C, and C \*/
9. parents := DNS parents of name
10. foreach dnsParent in parents
11. /\* Get all the objects that represent trusted domains \*/
12. tdos := select all o in Children ForestRootDomainNC() where
13. trustedDomain in o!objectClass and
14. o!trustAttributes & 0x00000008 ≠ 0 and
15. o!msDS-TrustForestTrustInfo ≠ null
16. foreach o in tdos
17. if not UnmarshalForestTrustInfo(o!msDS-TrustForestTrustInfo, f)
18. then
19. return false
20. else
21. foreach e in f.Entries
22. if (e.ForestTrustType = ForestTrustTopLevelName and
23. e.TopLevelName = dnsParent and
24. LSA\_FTRECORD\_DISABLED\_REASONS not in e.Flags) then
25. b := true
26. foreach g in f.Entries
27. if (g.ForestTrustType = ForestTrustTopLevelNameEx and
28. LSA\_FTRECORD\_DISABLED\_REASONS not in g.Flags and
29. (g.TopLevelName = dnsParent or
30. IsSubdomainOf(dnsParent, g.TopLevelName))) then
31. b := false
32. break
33. endif
34. endfor
35. if (b) then
36. trustedForestName := o!trustPartner
37. return true
38. endif
39. endif
40. endfor
41. endif
42. endfor
43. endfor
44. return false

## FOREST\_TRUST\_RECORD\_TYPE

FOREST\_TRUST\_RECORD\_TYPE is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for specifying the type of record contained in a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) trust information ([FOREST\_TRUST\_INFORMATION](#Section_642c0d174f3b4752a9a3464b080bf0b6)) entry. The allowed values are specified by the following enumerated list.

1. typedef enum
2. {
3. ForestTrustTopLevelName = 0,
4. ForestTrustTopLevelNameEx = 1,
5. ForestTrustDomainInfo = 2
6. } FOREST\_TRUST\_RECORD\_TYPE;

## ForestRootDomainNC

1. procedure ForestRootDomainNC(): DSName

The ForestRootDomainNC procedure returns the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the [**forest root domain NC**](#gt_9259fc5d-b976-44b0-b9a8-f7fe5e5ecf85).

## FullReplicaExists

1. procedure FullReplicaExists(nc : DSName) : boolean

The FullReplicaExists procedure returns true if the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) with root *nc* is a full [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac).

1. if not ObjExists(nc) then
2. return false
3. endif
4. return nc in (DSAObj()!msDS-hasMasterNCs +
5. DSAObj()!msDS-hasFullReplicaNCs)

## GCPAS

1. procedure GCPAS() : PARTIAL\_ATTR\_VECTOR\_V1\_EXT

*Informative summary of behavior*: The GCPAS procedure returns a reference to an instance of the [PARTIAL\_ATTR\_VECTOR\_V1\_EXT](#Section_1d5c1b34daa44761a8b5d3c0146a0e30) structure, which contains the list of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that can be present, based on the [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093), on a [**GC**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169) [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

1. partialAttrSetSeq: sequence of DSName
2. pPartialAttrVector: PARTIAL\_ATTR\_VECTOR\_V1\_EXT^
3. partialAttrSetSeq := select o from subtree SchemaNC() where
4. (o!isMemberOfPartialAttributeSet = true)
5. pPartialAttrVector = new PARTIAL\_ATTR\_VECTOR\_V1\_EXT sized to hold
6. partialAttrSetSeq.length entries in its rgPartialAttr
7. field
8. pPartialAttrVector^.dwVersion := 1
9. pPartialAttrVector^.cAttrs := partialAttrSetSeq.length
10. for i := 0 to partialAttrSetSeq.length-1
11. pPartialAttrVector^.rgPartialAttr[i]:=
12. AttrtypFromSchemaObj(partialAttrSetSeq[i])
13. endfor
14. return pPartialAttrVector

## GetFilteredAttributeSet

1. procedure GetFilteredAttributeSet() : sequence of ATTRTYP

*Informative summary of behavior*: The GetFilteredAttributeSet procedure returns a sequence of [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) that represents the list of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that cannot be present on a filtered [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210)

1. filteredAttrSet: sequence of ATTRTYP
2. filteredAttrSetObjSeq: sequence of DSName
3. i: int
4. filteredAttrSetObjSeq := select o from subtree SchemaNC() where
5. (fRODCFilteredAttribute in o!searchFlags) and
6. (not FLAG\_ATTR\_REQ\_PARTIAL\_SET\_MEMBER in
7. o!systemFlags) and
8. (not o!systemOnly = true) and
9. (not AttrtypFromSchemaObj(o) in
10. {currentValue, dBCSPwd, unicodePwd,
11. ntPwdHistory, priorValue,
12. supplementalCredentials, trustAuthIncoming,
13. trustAuthOutgoing, lmPwdHistory,
14. initialAuthIncoming, initialAuthOutgoing,
15. msDS-ExecuteScriptPassword, displayName,
16. codePage, creationTime, lockoutDuration,
17. lockOutObservationWindow, logonHours,
18. lockoutThreshold, maxPwdAge, minPwdAge,
19. minPwdLength, netbiosName, pwdProperties,
20. pwdHistoryLength, pwdLastSet,
21. securityIdentifier, trustDirection,
22. trustPartner, trustPosixOffset, trustType,
23. rid, domainReplica, accountExpires,
24. ntMixedDomain, OperatingSystem,
25. OperationSystemVersion,
26. operatingSystemServicePack, fsmoRoleOwner,
27. trustAttributes, trustParent, flatName,
28. sidHistory, dnsHostName, lockoutTime,
29. servicePrincipalName, isCriticalSystemObject,
30. msDS-TrustForestTrustInfo, msDS-SPNSuffixes,
31. msDS-AdditionalDnsHostName, msDS-
32. AdditionalSamAccountName, msDS-
33. AllowedToDelegateTo, msDS-KrbTgtLink, msDS-
34. AuthenticatedAtDC, msDS-
35. SupportedEncryptionTypes})
37. for i := 0 to filteredAttrSetObjSeq.length-1
38. filteredAttrSet[i]:=
39. AttrtypFromSchemaObj(filteredAttrSetObjSeq [i])
40. endfor
41. return filteredAttrSet

## GetNCType

1. procedure GetNCType(nc: DSName) : ULONG

*Informative summary of behavior*: The GetNCType procedure returns the type of the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

1. ncType: ULONG
2. ncType = 0;
3. if not AmIRODC() then
4. if not nc = ConfigNC() and
5. not nc = SchemaNC() and
6. not nc = DefaultNC() and
7. IsApplicationNC(nc) = false then
8. /\* the NC replica correspond to a GC partition \*/
9. ncType := ncType + {NCT\_GC\_PARTIAL}
10. endif
11. else if
12. if nc = ConfigNC() or
13. nc = DefaultNC() or
14. ApplicationNC(nc) = true then
15. ncType := ncType + {NCT\_FILTERED\_ATTRIBUTE\_SET,
16. NCT\_SPECIAL\_SECRET\_PROCESSING }
17. else if nc = SchemaNC() then
18. ncType := 0
19. else
20. ncType := ncType + {NCT\_FILTERED\_ATTRIBUTE\_SET,
21. NCT\_GC\_PARTIAL}
22. endif
23. endif
24. return ncType

## GetAttrVals

1. procedure GetAttrVals(
2. o: DSName,
3. att: ATTRTYP,
4. includeDeletedLinks: boolean): set of attribute value

The GetAttrVals procedure constructs a set V that contains each value of the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) *att* from [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *o*.

If *att* is not a [**link attribute**](#gt_be41074d-ce6b-4488-853a-4bbb3ea243ce), the value of *includeDeletedLinks* is ignored. If *att* is a link attribute and *includeDeletedLinks* = false, the set includes only those values *v* of *att* such that [LinkStamp](#Section_85075a8ff8a848d390e633ca806e31a7)(*o*, *att*, *v*).timeDeleted = 0. If *att* is a link attribute and *includeDeletedLinks* = true, the set contains all values *v* of *att*, even those such that LinkStamp(*o*, *att*, *v*).timeDeleted ≠ 0.

If the V is empty, null is returned. Otherwise, V is returned.

## GetCallerAuthorizationInfo

1. procedure GetCallerAuthorizationInfo(): ClientAuthorizationInfo

The GetCallerAuthorizationInfo procedure returns the [ClientAuthorizationInfo](#Section_92e47a548f244182b6b484f28699f8a1) (a security token) of the current caller. For more details, see [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.5.3.

## GetDefaultObjectCategory

1. procedure GetDefaultObjectCategory(class: ATTRTYP): DSName

The GetDefaultObjectCategory procedure returns the defaultObjectCategory value for the [**object class**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a) *class*.

1. classObj: DSName
2. classObj := SchemaObj(class)
3. return classObj!defaultObjectCategory

## GetDomainNC

1. procedure GetDomainNC(o: DSName): DSName

The GetDomainNC procedure returns one of the following:

* The [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) in which the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) whose DSName is *o* is located if *o* is in an application partition.
* The DSName of the NC that is the root for the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) where *o* is located if *o* is in a domain partition.
* NULL if the NC is not found or it is not of the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) form specified in [[RFC2253]](https://go.microsoft.com/fwlink/?LinkId=90327).

## GetDSNameFromAttrVal

1. procedure GetDSNameFromAttrVal(attrTyp: ATTRTYP, attrVal: ATTRVAL): DSName

The GetDSNameFromAttrVal procedure extracts the [**DSName**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a) from the [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) *attrVal* based on its syntax, which is determined from the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) *attrTyp*. If the syntax is not one of Object(DS-DN), Object(DN-String), or Object(DN-Binary), this procedure returns a null DSName.

## GetDSNameFromDN

1. procedure GetDSNameFromDN(dn: unicodestring): DSName

The GetDSNameFromDN procedure produces a [DSName](#Section_a0d5477a522946b9890a54b924d487d1) from *dn*. Let *d* represent the returned DSName. It is the case that d.dn = *dn*. If there is an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *o* in an [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) hosted by the server such that *o*!distinguishedName = *dn*, then d.guid =*o*!objectGUID; otherwise, all fields of d.guid are zero. Furthermore, if *o*!objectSid ≠ null, then d.sid = *o*!o[objectSid](%5bMS-ADA3%5d.docx#Section_afac8414c6144c6ab31641f5978308bd); otherwise d.sid = null.

## GetDSNameFromNetworkAddress

1. procedure GetDSNameFromNetworkAddress(n: NetworkAddress): DSName

The GetDSNameFromNetworkAddress procedure produces a [DSName](#Section_a0d5477a522946b9890a54b924d487d1) from the [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) *n*. If *n* is an [**FQDN (1)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) such that, for a [**DSA object**](#gt_dc90b593-841f-4c6d-8032-b32e58e887a8) *d*, there is a [**server object**](#gt_62a8c543-5998-480b-8fa7-41a8f04a18e5) *s* such that *d*!parent = *s* and *s*!dnsHostName = *n*, then return the [**DSName**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a) of *d*. If *n* is in the format <DC-name>:<DC-identifier> as described in section 5.134, return the DSName of the DC's DSA object. Otherwise return NULL.

## GetForestFunctionalLevel

1. procedure GetForestFunctionalLevel(): integer

The GetForestFunctionalLevel procedure returns the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) functional level (see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.4.4).

1. partitionsContainer: DSName
2. partitionsContainer:= DescendantObject(ConfigNC(), "CN=Partitions,")
3. if partitionsContainer!msDS-Behavior-Version = null then
4. return DS\_BEHAVIOR\_WIN2000
5. else
6. return partitionsContainer!msDS-Behavior-Version
7. endif

## GetFSMORoleOwner

1. procedure GetFSMORoleOwner(role: integer): DSName

The GetFSMORoleOwner procedure returns the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that owns the [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f) specified by *role*. The following table lists the valid values for *role*.

| Symbolic constant | Value |
| --- | --- |
| FSMO\_SCHEMA | 1 |
| FSMO\_DOMAIN\_NAMING | 2 |
| FSMO\_PDC | 3 |
| FSMO\_RID | 4 |
| FSMO\_INFRASTRUCTURE | 5 |

## GetInstanceNameFromSPN

1. procedure GetInstanceNameFromSPN(spn: unicodestring): unicodestring

The GetInstanceNameFromSPN procedure syntactically extracts and returns the instance name from a two-part or three-part [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4). The instance name is the second part of the SPN. For example, dc-01.fabrikam.com is the instance name in the two-part SPN "ldap/dc-01.fabrikam.com" and in the three-part SPN "ldap/dc-01.fabrikam.com/fabrikam.com".

## GetObjectNC

1. procedure GetObjectNC(o: DSName): DSName

The GetObjectNC procedure returns the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) in which the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) whose DSName is *o* is located, or returns NULL if the NC is not found or it is not of the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) form specified in [[RFC2253]](https://go.microsoft.com/fwlink/?LinkId=90327).

## GetProxyEpoch

1. procedure GetProxyEpoch(dnbinValue: DNBinary): DWORD

The GetProxyEpoch procedure returns the decoded proxy epoch field from the *dnbinValue*, which is a proxiedObjectName value.

## GetProxyType

1. procedure GetProxyType(dnbinValue: DNBinary): DWORD

The GetProxyType procedure returns the decoded proxy type field from the *dnbinValue*, which is a proxiedObjectName value.

## GetServiceClassFromSPN

1. procedure GetServiceClassFromSPN(spn: unicodestring): unicodestring

The GetServiceClassFromSPN procedure syntactically extracts and returns the [**service class**](#gt_c1386afd-9d42-47c7-a7ea-d01912a17647) from a two-part or three-part [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4). The service class is the first part of the SPN. For example, "ldap" is the service class in the two-part SPN "ldap/dc-01.fabrikam.com" and in the three-part SPN "ldap/dc-01.fabrikam.com/fabrikam.com".

## GetServiceNameFromSPN

1. procedure GetServiceNameFromSPN(spn: unicodestring): unicodestring

The GetServiceNameFromSPN procedure syntactically extracts and returns the service name from a three-part [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4). If the supplied SPN is a two-part SPN, it will return null. The service name is the third part of the SPN. For example, "fabrikam.com" is the service name in the three-part SPN "ldap/dc-01.fabrikam.com/fabrikam.com".

## groupType Bit Flags

The groupType bit flags can appear in values of the groupType [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that define a group type. The bit flags are presented below in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | Q G | B G | U G | R G | A G | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | S E | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**AG (GROUP\_TYPE\_ACCOUNT\_GROUP, 0x00000002)**: The account [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) type.

**RG (GROUP\_TYPE\_RESOURCE\_GROUP, 0x00000004)**: The resource group type.

**UG (GROUP\_TYPE\_UNIVERSAL\_GROUP, 0x00000008)**: The [**universal group**](#gt_f46053d6-0708-4094-ac63-57c1bcb73d32) type.

**BG (GROUP\_TYPE\_APP\_BASIC\_GROUP, 0x00000010)**: The application basic group type.

**QG (GROUP\_TYPE\_APP\_QUERY\_GROUP, 0x00000020)**: The application query group type.

**SE (GROUP\_TYPE\_SECURITY\_ENABLED, 0x80000000)**: The group is security-enabled.

## GUID

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d), as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) and [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.3.4.

The type [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) has a well-defined null value, which is called the [**NULL GUID**](#gt_ba500a5b-8c29-467c-a335-0980c8b11304). The constant [NULLGUID](#Section_61ffe2b7006f4bdeaf6cc3ecb8998eb9) is equal to this value.

When comparing two GUID values, each GUID value is treated as an octet string in little-endian byte order.

Two GUID values g1 and g2 are equal if they are octet-for-octet identical.

Value g1 is less than value g2 only if there exists an N (where N is less than the size of the GUID type in octets) such that octets 0...N-1 of g1 and g2 are identical, and octet N of g1 is less than octet N of g2.

Value g1 is greater than value g2 only if there exists an N (where N is less than the size of the GUID type in octets) such that octets 0...N-1 of g1 and g2 are identical, and octet N of g1 is greater than octet N of g2.

## GuidFromString

1. procedure GuidFromString(BracedFormat: boolean,
2. strGuid: unicodestring): GUID

The GuidFromString procedure converts the string representation of a [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) specified in *strGuid* (for example, "{12AA5F43-C776-4D63-B347-1175DF806200}" or "12aa5f43-c776-4d63-b347-1175df806200") to a binary GUID. When *BracedFormat* is true, to be a valid string representation of a GUID, *strGuid* MUST be in the curly braced GUID string format as defined in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.3.4.3. When *BraceFormat* is false, to be a valid string representation of a GUID, *strGuid* MUST be in the string GUID format as defined in [[RFC4122]](https://go.microsoft.com/fwlink/?LinkId=90460). If *strGuid* is not a valid string representation of a GUID, null is returned.

## GuidToString

1. procedure GuidToString(guid: GUID): unicodestring

The GuidToString procedure converts *guid* to the concatenation of "{", the string representation defined in [[RFC4122]](https://go.microsoft.com/fwlink/?LinkId=90460) section 3, and "}"; for example, {12aa5f43-c776-4d63-b347-1175df806200}.

## handle\_t

handle\_t is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle, as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 4.2.9.7 and [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.1.3.

## instanceType Bit Flags

The instanceType bit flags are bits that can appear in values of the instanceType [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f). The bit flags are presented in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | N G | N C | N A | W R | U I | N H | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**NH (IT\_NC\_HEAD, 0x00000001)**: The [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) is the root of an [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942).

**UI (IT\_UNINSTANT, 0x00000002)**: The [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) has not yet been instantiated.

**WR (IT\_WRITE, 0x00000004)**: The object is writable.

**NA (IT\_NC\_ABOVE, 0x00000008)**: The [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) hosts the NC above this one. The IT\_NC\_HEAD bit must also be set.

**NC (IT\_NC\_COMING, 0x00000010)**: The NC replica is in the process of being constructed for the first time via [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb). IT\_NC\_HEAD must also be set.

**NG (IT\_NC\_GOING, 0x00000020)**: The NC replica is in the process of being removed from the DC. IT\_NC\_HEAD must also be set.

## Is2PartSPN

1. procedure Is2PartSPN(spn: unicodestring): boolean

The Is2PartSPN procedure returns true if *spn* is an [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) consisting of two parts, and false otherwise.

## Is3PartSPN

1. procedure Is3PartSPN(spn: unicodestring): boolean

The Is3PartSPN procedure returns true if *spn* is an [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) consisting of three parts, and false otherwise.

## IsAdlds

1. procedure IsAdlds() : boolean

If the [**local DC**](#gt_17b69a5a-adc1-4763-92cf-5e44f11abbe7) is running [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) as [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab), this procedure returns TRUE. It returns FALSE otherwise.

## IsBuiltinPrincipal

1. procedure IsBuiltinPrincipal(sid: SID): boolean

The IsBuiltinPrincipal procedure returns true if *sid* is the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of a [**built-in security principal**](#gt_232336e6-3c6a-4c6b-8c35-52bbcf3c090a), and returns false if it is not.

## IsDomainNameInTrustedForest

1. procedure IsDomainNameInTrustedForest(name: unicodestring,
2. var referredDomain: unicodestring) : boolean

The IsDomainNameInTrustedForest procedure returns true if the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) identified by *name* is in a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) trusted by the caller's forest, as determined by the [FOREST\_TRUST\_INFORMATION](#Section_642c0d174f3b4752a9a3464b080bf0b6) state of the caller's forest, and false otherwise. The *name* parameter can be either an [**FQDN (1)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) or a NetBIOS name of a domain. If the IsDomainNameInTrustedForest procedure returns true, the *referredDomain* parameter value will be set to the FQDN (1) of the root domain of the forest of the domain specified by the *name* parameter. If the IsDomainNameInTrustedForest procedure returns false, the value of the *referredDomain* parameter remains unchanged.

See section 5.64 for the specification of this procedure.

## IsDomainSidInTrustedForest

1. procedure IsDomainSidInTrustedForest(sid: SID): boolean

The IsDomainSidInTrustedForest procedure returns true if the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) identified by *sid* is in a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) trusted by the caller's forest, as determined by the [FOREST\_TRUST\_INFORMATION](#Section_642c0d174f3b4752a9a3464b080bf0b6) state of the caller's forest, and false otherwise. The *sid* parameter is the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of a domain.

See section [5.64.2](#Section_f6bf681d6dc8407a8b0b3f0bd5fcee26) for the specification of this procedure.

## IsDCAccount

1. procedure IsDCAccount(o: DSName): boolean

The IsDCAccount procedure returns true if the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) represents the computer account of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

## IsForwardLinkAttribute

1. procedure IsForwardLinkAttribute (att:ATTYTYP): Boolean

The IsForwardLinkAttribute procedure returns true if the given [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) represents a [**forward link attribute**](#gt_ca910b1e-dfb2-4a06-94a8-425013020fb9). Forward link attribute is defined in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.1.6.

## IsGC

1. procedure IsGC(): boolean

The IsGC procedure returns true if the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) on which it is called is a [**global catalog server**](#gt_a5a99ce4-e206-42dc-8874-e103934c5b0d) as defined in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.1.10 or is in a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) that contains only one [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). Otherwise, the procedure returns false.

## IsGetNCChangesPermissionGranted

1. procedure IsGetNCChangesPermissionGranted(
2. msgIn: DRS\_MSG\_GETCHGREQ\_V10) : boolean

*Informative summary of behavior*: The IsGetNCChangesPermissionGranted procedure returns true if the source [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) has permission to replicate [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) and its [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) from the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210), as defined in *msgIn*.

1. ncRoot: DSName
2. clientDsaObj: DSName
3. serverObj: DSName
4. cachedAt: DSName
5. cachedUser: DSName
6. fRevealSecret: boolean
7. fRevealFilteredSet: boolean
8. ncRoot := GetObjectNC(msgIn.pNC^)
9. if not AccessCheckCAR(ncRoot, Ds-Replication-Get-Changes) then
10. return false
11. endif
12. fRevealSecret := true
13. if IsRevealSecretRequest(msgIn) then
14. if AccessCheckCAR(ncRoot, Ds-Replication-Get-Changes-All) = false
15. then
16. if (msgIn.ulExtendedOp = EXOP\_REPL\_SECRETS) then
17. clientDsaObj := select one o from ConfigNC()where
18. o!objectGUID = msgIn.uuidDsaObjDest
19. serverObj := clientDsaObj!parent
20. cachedAt := serverObj!serverReference
21. cachedUser := msgIn.pNC^
22. fRevealSecret := RevealSecretsForUserAllowed(
23. cachedAt, cachedUser)
24. else
25. fRevealSecret := false
26. endif
27. endif
28. endif
29. fRevealFilteredSet := true
30. if IsRevealFilteredAttr(msgIn) then
31. if (AccessCheckCAR(ncRoot, Ds-Replication-Get-Changes-All) = false
32. and
33. AccessCheckCAR(ncRoot, Ds-Replication-Get-Changes-In-Filtered-Set)
34. = false) then
35. fRevealFilteredSet := false
36. endif
37. endif
38. if (fRevealSecret = false) or (fRevealFilteredSet = false)
39. return false
40. else
41. return true
42. endif

## IsGUIDBasedDNSName

1. procedure IsGUIDBasedDNSName(o: DSName, instanceName: unicodestring):
2. boolean

The IsGUIDBasedDNSName procedure returns true if *instanceName* is the DNS host name of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), identified by *o*, constructed in the form "<DSA GUID>.\_msdcs.<DNS forest name>".

## IsMemberOfBuiltinAdminGroup

1. procedure IsMemberOfBuiltinAdminGroup(): boolean

The IsMemberOfBuiltinAdminGroup procedure returns true if the client [**security context**](#gt_88d49f20-6c95-4b64-a52c-c3eca2fe5709), which MUST be retrieved using the method described in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.3.4.3, is a member of the BUILTIN\Administrators [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac), and false if it is not. The BUILTIN\Administrators group is the group with the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) S-1-5-32-544. [[MS-SAMR]](%5bMS-SAMR%5d.pdf#Section_4df07fab1bbc452f8e927853a3c7e380) section 3.1.4.2 describes the accounts included in the built-in Administrators group by default.

## IsRecycleBinEnabled

1. procedure IsRecycleBinEnabled(): boolean

*Informative summary of behavior*: The IsRecycledBinEnabled procedure returns true if the [Recycle Bin](#gt_54624800-58f4-45e9-90bf-c9b52dcf98f3) [optional feature](#gt_785b66f1-22b3-450f-97aa-a24a39d04d47) is enabled. Otherwise, it returns false. For more details, see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) sections 3.1.1.9, 3.1.1.9.1, and 6.1.1.2.1

1. recycleBinFeatureGuid: GUID
2. scope: DSNAME
3. recycleBinFeatureGuid := {766ddcd8-acd0-445e-f3b9-a7f9b6744f2a}
4. scope := DSName of the nTDSDSA object
5. return IsOptionalFeatureEnabled (scope, recycleBinFeatureGuid)

## IsRevealFilteredAttribute

1. procedure IsRevealFilteredAttribute(
2. DRS\_MSG\_GETCHGREQ\_V10 msgIn) : boolean

*Informative summary of behavior*: The IsRevealFilteredAttribute procedure returns true if the source [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) is requesting [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) in the filtered set.

1. filteredAttrSet: sequence of ATTRTYP
2. i: int
3. filteredAttrSet := GetFilteredAttributeSet()
4. for i := 0 to msgIn.pPartialAttrSet.cAttrs - 1
5. if msgIn.pPartialAttrSet.rgPartialAttr[i]) in
6. filteredAttrSet then
7. return true;
8. endif
9. endfor
10. for i := 0 to msgIn.pPartialAttrSetEx.cAttrs - 1
11. if msgIn.pPartialAttrSetEx.rgPartialAttr[i]) in
12. filteredAttrSet then
13. return true;
14. endif
15. endfor
16. return false;

## IsPrivilegedAccessManagementEnabled

1. procedure IsPrivilegedAccessManagementEnabled(): boolean

Informative summary of behavior: The IsPrivilegedAccessManagementEnabled procedure returns true if the [**Privileged Access Management**](#gt_2cc7e2f1-0f12-4357-9846-e772aef4de39) [**optional feature**](#gt_785b66f1-22b3-450f-97aa-a24a39d04d47) is enabled. Otherwise, it returns false. For more information, see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) sections 3.1.1.9, 3.1.1.9.2, and 6.1.1.2.1.

1. privilegedAccessManagementFeatureGuid: GUID
2. scope: DSNAME
3. privilegedAccessManagementFeatureGuid := {ec43e873-cce8-4640-b4ab-07ffe4ab5bcd}
4. scope := DSName of the nTDSDSA object
5. return IsOptionalFeatureEnabled (scope, privilegedAccessManagementFeatureGuid)

## IsRevealSecretRequest

1. procedure IsRevealSecretRequest(DRS\_MSG\_GETCHGREQ\_V10 msgIn)
2. : boolean

*Informative summary of behavior*: The IsRevealSecretRequest procedure returns true if the source [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) is requesting secret [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).

1. if AmILHServer() = false then
2. if (DRS\_WRITE\_REP in msgIn.ulFlags) then
3. return true
4. else
5. return false
6. endif
7. /\* if source DC is requesting FSMO related operation then it is same
8. as a reveal secret request \*/
9. if (msgIn.ulExtendedOp = EXOP\_FSMO\_REQ\_ROLE or
10. msgIn.ulExtendedOp = EXOP\_FSMO\_REQ\_RID\_ALLOC or
11. msgIn.ulExtendedOp = EXOP\_FSMO\_RID\_REQ\_ROLE or
12. msgIn.ulExtendedOp = EXOP\_FSMO\_REQ\_PDC or
13. msgIn.ulExtendedOp = EXOP\_FSMO\_ABANDON\_ROLE) then
14. return true
15. endif
16. /\* if source DC is requesting for special secrets processing then it
17. implies that it is not requesting for secrets \*/
18. if ({DRS\_SPECIAL\_SECRET\_PROCESSING} ∩ msgIn.ulFlags) then
19. return false
20. endif
21. if (msgIn.ulExtendedOp = EXOP\_REPL\_SECRETS or
22. msgIn.pAttributeSet = null then /\* requesting all attributes that
23. includes secrets\*/
24. return true
25. endif
26. for i := 0 to msgIn.pPartialAttrSet.cAttrs - 1
27. if IsSecretAttribute(msgIn.pPartialAttrSet.rgPartialAttr[i]) then
28. return true;
29. endif
30. endfor
31. for i := 0 to msgIn.pPartialAttrSetEx.cAttrs - 1
32. if IsSecretAttribute(msgIn.pPartialAttrSetEx.rgPartialAttr[i]) then
33. return true;
34. endif
35. endfor
36. return false;

## IsServerExtensionsChanged

1. procedure IsServerExtensionsChanged(
2. ServerExtensions: DRS\_EXTENSIONS\_INT): boolean;

The IsServerExtensionsChanged procedure returns true if the supplied extensions are different from the current server extensions. Otherwise, it returns false.

## IsUPNInTrustedForest

1. prodecure IsUPNInTrustedForest(upn: unicodestring): boolean

The IsUPNInTrustedForest procedure returns true if the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) containing the account identified by *upn* is in a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) trusted by the caller's forest, as determined by the [FOREST\_TRUST\_INFORMATION](#Section_642c0d174f3b4752a9a3464b080bf0b6) state of the caller's forest, and false otherwise. The *upn* parameter is the UPN of an account in a domain.

See section 5.64 for the specification of this procedure.

## IsValidServiceName

1. procedure IsValidServiceName(o: DSName, serviceName: unicodestring):
2. boolean

The IsValidServiceName procedure returns true if the name *serviceName* is a valid service name in an [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) for the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) represented by computer [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *o*.

A valid service name can be one of the following:

1. For [**GC**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169) SPNs, the service name must be the DNS [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) name.
2. For other [**classes**](#gt_18393bbe-0c06-42b7-890d-b94a9a40b6e0) of SPNs, the service name must be either the DNS [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) name of the DC's default domain or the DNS domain name of an [**application NC**](#gt_53fee475-b07f-45e1-b4b7-c7ac0c1e7f6a) hosted by the DC.

## KCCFailedConnections

KCCFailedConnections is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) consisting of a sequence of tuples, one tuple for each [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) for which the connection attempt failed. Each tuple contains the following fields:

* **DsaDN**: A *unicodestring* (section [3.4.3](#Section_fbe9988847824858b5f25b521a44d836)) that contains the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that corresponds to the DC.
* **UUIDDsa**: A [GUID](#Section_5e740f50e6a048c9bca800072e85d963) that contains the [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the DC.
* **TimeFirstFailure**: A [FILETIME](#Section_70ee934bc9b944498aa36dfe9cef3eff) that contains the time when the [**KCC**](#gt_c7d4f1f6-5285-4168-b21a-022f775a3f58) noticed the first failure while contacting the DC.
* **FailureCount**: An integer that contains the total number of failures the KCC encountered while contacting the DC.
* **LastResult**: A [DWORD](#Section_60c3f5f194924d1083c89a155e162ef3) that contains a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b), as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.2, that indicates the reason for the last failure.

The global variable [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219) for a DC has an associated field dc.kccFailedConnections, which maintains the DC's KCCFailedConnections state.

## KCCFailedLinks

KCCFailedLinks is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that consists of a sequence of tuples, one tuple for each neighboring [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) for which a connection attempt failed.

The fields of the tuple are the same as the fields of the [KCCFailedConnections](#Section_eaffa80d8baf4784898ee9fbc7bd8296) tuple.

The global variable [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219) for a DC has an associated field dc.kccFailedLinks, which maintains the DC's KCCFailedLinks state.

## LARGE\_INTEGER

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a 64-bit signed integer, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.3.5.

## LDAP\_CONN\_PROPERTIES

LDAP\_CONN\_PROPERTIES is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) that contains bit flags that identify properties of an [**LDAP connection**](#gt_198f4791-cea3-465d-89e2-262991624e08). The bit flags are presented below in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| M D 5 | S P L | N G O | G S S | G C | U D P | S S L | B N D | X | X | X | X | X | X | S L | S G N | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**BND (0x00000001)**: A bind has been performed on this LDAP connection.

**SSL (0x00000002)**: The LDAP connection corresponds to a Secure Sockets Layer (SSL) connection.

**UDP (0x00000004)**: The LDAP connection corresponds to a User Datagram Protocol (UDP) connection.

**GC (0x00000008)**: The LDAP connection was made through the [**GC**](#gt_4f5d605a-7b3f-4db7-8c21-b146856d7169) port.

**GSS (0x00000010)**: The Generic Security Services Application Programming Interface (GSS-API) security package was used for [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317).

**NGO (0x00000020)**: The Simple and Protected GSS-API Negotiation (SPNEGO) security package was used for authentication.

**SPL (0x00000040)**: The [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) bind corresponds to LDAP simple bind.

**MD5 (0x00000080)**: The Digest-MD5 security package was used for authentication.

**SGN (0x00000100)**: Signing is enabled on the LDAP connection.

**SL (0x00000200)**: Sealing is enabled on the LDAP connection.

## LDAP\_SERVER\_DIRSYNC\_OID LDAP Search Control

This section describes LDAP\_SERVER\_DIRSYNC\_OID control processing. See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.3.4.1.3 for more details.

### Abstract Types

#### AttributeList

1. Type AttributeList = [next: AttributeList,
2. value: AttributeListElement]

AttributeList is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that contains information about the [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). It is a tuple of the following:

**next**: The next tuple.

**value**: An attribute type and its value(s).

#### AttributeListElement

1. Type AttributeListElement = [type: LDAPString,
2. vals: AttributeVals]

AttributeListElement is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that contains information about an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). It is a tuple of the following:

**type**: Attribute type.

**vals**: Attribute values.

#### AttributeVals

1. Type AttributeVals = [next: AttributeVals,
2. value: LDAPString]

AttributeVals is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that contains information about the value(s) of an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f). It is a tuple of the following:

**next**: The next tuple.

**value**: An attribute value.

#### Control

1. Type Control = [controlType: LDAPString, criticality: BOOL,
2. controlValue: LDAPString]

Control is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that contains information about an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) control. It is a tuple of the following:

**controlType**: LDAP control [**OID**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2).

**criticality**: LDAP control criticality.

**controlValue**: LDAP control value.

#### DirSyncControlValue

1. Type DirSyncControlValue = [flags: ULONG, size: ULONG, cookie: LDAPString]

DirSyncControlValue is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that contains information about the LDAP\_SERVER\_DIRSYNC\_OID control value. It is a tuple of the following:

**flags**: Flags, as described in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.3.4.1.3.

**size**: In the request control value, this field indicates the maximum number of bytes expected in the reply. In the reply control value, it is set to 0.

**cookie**: A cookie value.

#### DirSyncSearchArg

1. Type DirSyncSearchArg = [pObject: LDAPString, pFilter: LDAPString,
2. pSelection: ATTRBLOCK, sizeLimit: ULONG]

DirSyncSearchArg is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that contains information about the [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) search request. It is a tuple of the following:

**pObject**: The LDAP search base [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b).

**pFilter**: The string representation, as defined in [[RFC2254]](https://go.microsoft.com/fwlink/?LinkId=90328), of the LDAP search filter.

**pSelection**: The [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) selection info in the LDAP search request.

**sizelimit**: The size limit in the LDAP search request.

#### LDAPString

1. Type LDAPString = [length: ULONG, value: \* UCHAR]

LDAPString is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that contains information about a string of unsigned characters. It is a tuple of the following:

**length**: The number of unsigned characters in **value**.

**value**: The string of unsigned characters that can contain null characters.

#### SearchResultEntry

1. Type SearchResultEntry = [objectName: LDAPString,
2. attributes: AttributeList]

SearchResultEntry is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that contains information about a search result entry. It is a tuple of the following:

**objectName**: The [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b).

**attributes**: The list of the [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of object.

#### SearchResultEntryList

1. Type SearchResultEntryList = [next: SearchResultEntryList,
2. entry: SearchResultEntry]

SearchResultEntryList is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that contains information about search result entries. It is a tuple of the following:

**next**: The next tuple.

**entry**: An entry in the search result.

### Concrete Types

#### Cookie

The Cookie structure is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) that contains information about the cookie in the LDAP\_SERVER\_DIRSYNC\_OID control value.

1. typedef struct {
2. UCHAR signature[4];
3. ULONG version;
4. FILETIME creationTime;
5. ULONGLONG reserved;
6. ULONG utdVectorSize;
7. USN\_VECTOR usnVector;
8. UUID uuidSourceDsaInvocationID;
9. UCHAR utdVector[variable];
10. } Cookie;

**signature:**  Cookie signature.

**version:**  The version number.

**creationTime:**  The creation time.

**reserved:**  Unused.

**utdVectorSize:**  The [**up-to-date vector**](#gt_42564a26-2ae7-41a2-a67c-3c74381d8538) size.

**usnVector:**  The [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) vector.

**uuidSourceDsaInvocationID:**  The [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) (a [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3)) of the source DSA.

**utdVector:**  The up-to-date vector.

### ProcessDirSyncSearchRequest

1. ProcessDirSyncSearchRequest(
2. [in] searchArg: DirSyncSearchArg,
3. [in] dirSyncControlValue: DirSyncControlValue,
4. [out] searchResultEntryList: SearchResultEntryList,
5. [out] dirSyncResponseControl: Control
6. ) : ULONG

*Informative summary of behavior*: The ProcessDirSyncSearchRequest procedure processes an [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) search request with an LDAP\_SERVER\_DIRSYNC\_OID control. It creates a list of search result entries and the LDAP\_SERVER\_DIRSYNC\_OID response control.

1. err: ULONG
2. msgIn: DRS\_MSG\_GETCHGREQ\_V10
3. msgOut: DRS\_MSG\_GETCHGREPLY\_NATIVE
4. filter: LDAPString
5. replFlags: ULONG
6. /\* Transform the LDAP search request with LDAP\_SERVER\_DIRSYNC\_OID control to
7. a replication GetChanges request. \*/
8. err := DirSyncReqToGetChgReq(searchArg, dirSyncControlValue, msgIn)
9. if err ≠ 0 then
10. return err
11. endif
12. replFlags := dirSyncControlValue.flags
13. /\* Perform access checks unless client has specified object-level security \*/
14. if not (LDAP\_DIRSYNC\_OBJECT\_SECURITY in replFlags) then
15. err := SecurityCheckForChanges(msgIn)
16. if err ≠ 0 then
17. return err
18. endif
19. endif
20. filter := searchArg.pfilter
21. /\* Perform normal replication (Get replication changes). \*/
22. err := GetReplChanges(null, filter, replFlags, msgIn, msgOut)
23. if err ≠ 0 then
24. return err
25. endif
26. /\* Transform the replication GetChanges reply to reply for the LDAP search
27. request with LDAP\_SERVER\_DIRSYNC\_OID control. \*/
28. err := GetChgReplyToSearchResult(msgOut, searchResultEntryList,
29. dirSyncResponseControl)
30. if err ≠ 0 then
31. return err
32. endif
33. return 0 /\* success \*/

### DirSyncReqToGetChgReq

1. procedure DirSyncReqToGetChgReq(
2. [in] searchArg: DirSyncSearchArg,
3. [in] dirSyncControlValue: DirSyncControlValue,
4. [out] nativeRequest: DRS\_MSG\_GETCHGREQ\_V10
5. ): ULONG

*Informative summary of behavior*: The DirSyncReqToGetChgReq procedure transforms the received [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) search request with an LDAP\_SERVER\_DIRSYNC\_OID control message into a [DRS\_MSG\_GETCHGREQ\_V10](#Section_92b1b77d205846e09e8c6664b96a0cf9) request.

1. baseObjectDsName: DSNAME
2. replFlags: ULONG
3. ulSizeLimit: ULONG
4. ulMaxBytes: ULONG
5. errCode: ULONG
6. /\* Validate input \*/
7. /\* If the base of the search is not the root of an NC,
8. return unwillingToPerform \*/
9. baseObjectDsName := GetDSNameFromDN (searchArg.pObject)
10. if baseObjectDsName ≠ GetObjectNC(baseObjectDsName) then
11. return unwillingToPerform
12. endif
13. nativeRequest.pNC := baseObjectDsName
14. nativeRequest.ulFlags := 0
15. nativeRequest.ulMoreFlags := 0
16. replFlags := dirSyncControlValue.flags
17. if (LDAP\_DIRSYNC\_ANCESTORS\_FIRST\_ORDER in replFlags) then
18. nativeRequest.ulFlags := nativeRequest.ulFlags + {DRS\_GET\_ANC}
19. endif
20. ulSizeLimit := 1000
21. if (searchArg.sizeLimit) then
22. ulSizeLimit := min(searchArg.sizeLimit, ulSizeLimit)
23. endif
24. nativeRequest.cMaxObjects = max (ulSizeLimit, 100)
25. ulMaxBytes := dirSyncControlValue.size
26. if (ulMaxBytes = 0) then
27. ulMaxBytes := 1024\*1024
28. endif
29. nativeRequest.cMaxBytes := ulMaxBytes
30. cookie := dirSyncControlValue.cookie
31. errCode := GetUsnUtdVectorFromCookie (cookie, nativeRequest)
32. if (errCode ≠ 0) then
33. return errCode
34. endif
35. /\* Handle attribute selection \*/
36. copy the list of required attributes, if present, from searchArg.pSelection
37. to nativeRequest.pPartialAttrSet
38. return 0 /\* success \*/

### GetChgReplyToSearchResult

1. procedure GetChgReplyToSearchResult (
2. [in] chgReply: DRS\_MSG\_GETCHGREPLY\_NATIVE,
3. [out] searchResultEntryList: SearchResultEntryList,
4. [out] dirSyncResponseControl: Control)

*Informative summary of behavior*: The GetChgReplyToSearchResult procedure generates a list of search result entries (*searchResultEntryList*) and an LDAP\_SERVER\_DIRSYNC\_OID response control (*dirSyncResponseControl*) from a DRS\_MSG\_GETCHGREPLY\_NATIVE structure.

The arguments to this procedure are as follows:

* *chgReply*: A DRS\_MSG\_GETCHGREPLY\_NATIVE message generated by the GetReplChanges function.
* *searchResultEntryList*: A list of search result entries.
* *dirSyncResponseControl*: The control that is sent back to the client.

1. objCount: ULONG
2. valCount: ULONG
3. replEntinfList: REPLENTINFLIST
4. responseControlValue: DirSyncControlValue
5. objectGuid: GUID
6. attrType: ATTRTYP
7. minCookieLength: ULONG
8. utdVectorSize: ULONG
9. objCount := chgReply.cNumObjects
10. valCount := chgReply.cNumValues
11. /\* Process object updates. \*/
12. replEntinfList := chgReply.pObjects
13. while (not replEntinfList = null)
14. TransformEntinfToSearchEntry(replEntinfList.Entinf,
15. searchResultEntryList.entry)
16. Add objectGUID attribute with value =
17. replEntinfList.Entinf.pName.Guid to
18. searchResultEntryList.entry
19. Add parentGUID attribute with value =
20. replEntinfList.pParentGuid to
21. searchResultEntryList.entry
22. replEntinfList := replEntinfList.pNextEntInf
23. searchResultEntryList := searchResultEntryList.next
24. endwhile
25. /\* Process value updates. \*/
26. foreach distinct object GUID objectGuid, referred by replValInf,
27. in chgReply.rgValues
28. TransformReplValInfNativeListToSearchEntry(
29. replValInf.pObject, chgReply, searchResultEntryList.entry)
30. Add objectGUID attribute with value =
31. replValInf.pObject.Guid to
32. searchResultEntryList.entry
33. searchResultEntryList := searchResultEntryList.next
34. endfor
35. /\* Construct LDAP\_SERVER\_DIRSYNC\_OID response control \*/
36. dirSyncResponseControl.value.controlType := LDAP\_SERVER\_DIRSYNC\_OID
37. dirSyncResponseControl.value.criticality := false
38. responseControlValue := GetResponseDirSyncControlValue (chgReply)
39. dirSyncResponseControl.value.controlValue :=
40. BER encoding of the responseControlValue
41. return

### TransformEntinfToSearchEntry

1. procedure TransformEntinfToSearchEntry (
2. [in] entinf: ENTINF
3. [out] searchResultEntry: SearchResultEntry
4. )

*Informative summary of behavior*: The TransformEntinfToSearchEntry procedure transforms an ENTINF structure (*entinf*) into a SearchResultEntry structure (*searchResultEntry*).

1. attrList: AttributeList
2. attrVals: AttributeVals
3. TransformDSNameToLdapDN (entinf.pName, searchResultEntry.objectName)
4. attrList := searchResultEntry.attributes
5. foreach i in [0 .. entInf.AttrBlock.attrCount -1] do
6. attrList.value.type := LDAPDisplayNameFromAttrTyp (
7. entInf.AttrBlock.pAttr[i].attrTyp)
8. attrVals := attrList.value.vals
9. foreach j in [0 .. entInf.AttrBlock. pAttr[i].AttrVal.valCount -1] do
10. attrVals.value := ValueFromATTRVAL(
11. entInf.AttrBlock.pAttr[i].AttrVal.pAVal[j],
12. Syntax(entInf.AttrBlock.pAttr[i].attrTyp),
13. dc.prefixTable)
14. attrVals := attrVals.next
15. endfor
16. attrList := attrList.next
17. endfor
18. return

### TransformReplValInfNativeListToSearchEntry

1. procedure TransformReplValInfNativeListToSearchEntry (
2. [in] o: DSNAME,
3. [in] chgReply: DRS\_MSG\_GETCHGREPLY\_NATIVE,
4. [out] searchResultEntry: SEARCH\_RESULT\_ENRTY
5. )

*Informative summary of behavior*: The TransformReplValInfNativeListToSearchEntry procedure transforms, for [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *o*, the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) values in the REPLVALINF\_NATIVE list (*chgReply*.**rgValues**) to an AttributeList structure in *searchResultEntry*.

1. attrList: AttributeList
2. attrVals: AttributeVals
3. attributeType: ATTRTYP
4. TransformDSNameToLdapDN (o, searchResultEntry.objectName)
5. attrList := searchResultEntry.attributes
6. foreach distinct attribute attrType of the object o in
7. chgReply.rgValues
8. attrList.value.type := LDAPDisplayNameFromAttrTyp (attrType)
9. attrVals := attrList.value.vals
10. foreach attribute value replAttrVal of the attribute attrType
11. of the object o in chgReply.rgValues
12. attrVals.value := ValueFromATTRVAL(replAttrVal,
13. attrType,
14. dc.prefixTable)
15. attrVals := attrVals.next
16. endfor
17. attrList := attrList.next
18. endfor
19. return

### TransformDSNameToLdapDN

1. procedure TransformDSNameToLdapDN (
2. [in] dsName: DSNAME
3. [out] dn: LDAPString
4. )

*Informative summary of behavior*: The TransformDSNameToLdapDN procedure transforms a DSNAME to an LDAPString [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b).

1. dn.length := dsName.NameLen
2. dn.value := dsName.StringName
3. return

### LDAPDisplayNameFromAttrTyp

1. Procedure LDAPDisplayNameFromAttrTyp (
2. [in] attrTyp: ATTRTYP
3. ) : LDAPString

*Informative summary of behavior*: The LDAPDisplayNameFromAttrTyp procedure transforms an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) type (ATTRTYP) to an attribute name that is used by the [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) clients.

1. oid : OID
2. attributeDisplayName: LDAPString
3. attrObj: DSName
4. oid := OidFromAttid(dc.prefixTable, attrTyp)
5. attrObj := select o from subtree SchemaNC() where
6. (attributeSchema in o!objectClass) and
7. (o!attributeID = oid)
8. attrDisplayname := attrObj!LDAPDisplayName
9. return attrDisplayName

### GetResponseDirSyncControlValue

1. procedure GetResponseDirSyncControlValue (
2. [in] chgReply: DRS\_MSG\_GETCHGREPLY\_NATIVE
3. ) : DirSyncControlValue

*Informative summary of behavior*: The GetResponseDirSyncControlValue procedure creates an LDAP\_SERVER\_DIRSYNC\_OID control value, to be returned in the response LDAP\_SERVER\_DIRSYNC\_OID control, from a DRS\_MSG\_GETCHGREPLY\_NATIVE structure (*chgReply*).

The arguments to this procedure are as follows:

* *chgReply*: A DRS\_MSG\_GETCHGREPLY\_NATIVE message generated by the GetReplChanges function.

1. replControlValue: DirSyncControlValue
2. minCookieLength: ULONG
3. utdVectorSize: ULONG
4. pUpToDateVecSrcV1: UPTODATE\_VECTOR\_V1\_EXT
5. /\* Construct LDAP\_SERVER\_DIRSYNC\_OID response control value \*/
6. replControlValue.flag := chgReply.fMoreData
7. replControlValue.size := 0 /\* must be ignored by the client \*/
8. /\* minimum possible cookie size in bytes; that is, the size of a cookie
9. when a UTD vector is not present in the cookie. \*/
10. minCookieLength := 17\*4
11. pUpToDateVecSrcV1 := If necessary, convert chgReply.pUpToDateVecSrc (of
12. type UPTODATE\_VECTOR\_V2\_EXT) to UPTODATE\_VECTOR\_V1\_EXT by
13. creating a new UPTODATE\_VECTOR\_V1\_EXT with a UPTODATE\_CURSOR\_V1
14. cursor for each UPTODATE\_CURSOR\_V2 cursor, ignoring the
15. timeLastSyncSuccess field in UPTODATE\_CURSOR\_V2.
16. utdVectorSize := 16 /\* offsetof(UPTODATE\_VECTOR,V1.rgCursors[0]) \*/
17. + (pUpToDateVecSrcV1.V1.cNumCursors \*
18. sizeof(UPTODATE\_CURSOR\_V1))
19. replControlValue.cookie.length := minCookieLength + utdVectorSize replControlValue.cookie.value.signature := "SDSM"
20. replControlValue.cookie.value.version := 3
21. replControlValue.cookie.value.creationTime := current system time
22. replControlValue.cookie.value.utdVectorSize := utdVectorSize
23. replControlValue.cookie.value.usnVector := chgReply.usnvecTo
24. replControlValue.cookie.value.uuidSourceDsaInvocationID :=
25. chgReply.uuidInvocIdSrc
26. copy utdVectorSize bytes from pUpToDateVecSrcV1 to
27. replControlValue.cookie.value.utdVector
28. return replControlValue

### GetUsnUtdVectorFromCookie

1. procedure GetUsnUtdVectorFromCookie(
2. [in] replCookie: LDAPString,
3. [in/out] nativeRequest: DRS\_MSG\_GETCHGREQ\_V10
4. ): ULONG

*Informative summary of behavior*: The GetUsnUtdVectorFromCookie procedure extracts the [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) vector and the UTD vector from an LDAP\_SERVER\_DIRSYNC\_OID control value (*replCookie*) and sets the USN vector **from** (*nativeRequest*.**usnvecFrom**) and UTD vector **destination** (*nativeRequest*.**pUpToDateVecDest**).

1. utdVectorSize: ULONG
2. minCookieLength: ULONG
3. /\* minimum possible cookie size in bytes; that is, the size of a cookie
4. when a UTD vector is not present in the cookie. \*/
5. minCookieLength := 17\*4
6. /\* Validate cookie, and extract USN and UTD vectors. \*/
7. If (replCookie.length ≠ 0) then
8. If (replCookie.length < minCookieLength)
9. or
10. (replCookie.value.signature ≠ "SDSM")
11. or
12. (replCookie.value.version ≠ 3)
13. return protocolError
14. endif
15. utdVectorSize := replCookie.value.utdVectorSize
16. if (utdVectorSize < sizeof(UPTODATE\_VECTOR\_V1\_EXT)
17. or
18. replCookie.length < minCookieLength +
19. utdVectorSize) then
20. utdVectorSize := 0
21. endif
22. if (replCookie.value.uuidSourceDsaInvocationId =
23. DSAObj()!invocationId) then
24. nativeRequest.usnvecFrom := replCookie.value.usnVector
25. endif
26. if (utdVectorSize > 0) then
27. Copy utdVectorSize bytes from replCookie.value.utdVector
28. to nativeRequest.pUpToDateVecDest
29. /\* some more validation \*/
30. if (nativeRequest.pUpToDateVecDest.dwVersion ≠ 1
31. or
32. 16 /\* (offsetof(UPTODATE\_VECTOR,V1.rgCursors[0]) \*/
33. + (nativeRequest.pUpToDateVecDest.V1.cNumCursors
34. \* sizeof(UPTODATE\_CURSOR\_V1))) ≠ utdVectorSize
35. or
36. replCookie.length ≠ minCookieLength
37. + utdVectorSize) then
38. return protocolError
39. endif
40. Endif
41. endif
42. return 0 /\* success \*/

### SecurityCheckForChanges

1. procedure SecurityCheckForChanges(
2. [in] msgIn: DRS\_MSG\_GETCHGREQ\_V10
3. ): ULONG

*Informative summary of behavior*: The SecurityCheckForChanges procedure checks whether an LDAP\_SERVER\_DIRSYNC\_OID control client has access rights to read the changes in an [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) (*msgIn*.**pNC**).

1. if AccessCheckCAR(msgIn.pNC, Ds-Replication-Get-Changes) = false then
2. return insufficientAccessRights
3. endif
4. if msgIn.pPartialAttrSet.cAttrs ≠ 0 and
5. IsFilteredAttributePresent(msgIn.pPartialAttrSet) = true and
6. AccessCheckCAR(msgIn.pNC,
7. Ds-Replication-Get-Changes-In-Filtered-Set) = false and
8. AccessCheckCAR(msgIn.pNC,
9. Ds-Replication-Get-Changes-All) = false
10. then
11. return insufficientAccessRights
12. endif
13. return 0 /\* success \*/

### IsFilteredAttributePresent

1. procedure IsFilteredAttributePresent(
2. [in] attrVec: PARTIAL\_ATTR\_VECTOR\_V1\_EXT
3. ) : boolean

*Informative summary of behavior*: The IsFilteredAttributePresent procedure returns true if an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) from the filtered set is present in *attrVec*; otherwise, it returns false.

1. filteredAttrSet: sequence of ATTRTYP
2. i: int
3. filteredAttrSet := GetFilteredAttributeSet()
4. for i := 0 to attrVec.cAttrs - 1
5. if attrVec.rgPartialAttr[i] in filteredAttrSet then
6. return true
7. endif
8. endfor
9. return false

## LDAPConnections

LDAPConnections is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) for the [**LDAP connections**](#gt_198f4791-cea3-465d-89e2-262991624e08) associated with a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). It is a sequence of tuples, one tuple per LDAP connection currently open. Each tuple contains the following fields:

* **iPAddress**: A [DWORD](#Section_60c3f5f194924d1083c89a155e162ef3) that contains the IPv4 address of the client machine that established the connection.
* **notificationCount**: An integer that contains the number of [**LDAP**](#gt_45643bfb-b4c4-432c-a10f-b98790063f8d) notifications enabled on the connection.
* **secTimeConnected**: An integer that contains the time, in seconds, that the connection has been open.
* **flags**: A DWORD that contains the [LDAP\_CONN\_PROPERTIES](#Section_09a9cd41caed441da7515a992800a4fb) bit flags that identify properties of the connection.
* **totalRequests**: An integer that contains the total number of LDAP requests processed on the connection.
* **userName**: A *unicodestring* (section [3.4.3](#Section_fbe9988847824858b5f25b521a44d836)) that contains the name of the [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) that opened the connection.
* **fschemaUpgradeInProgress**: A Boolean that specifies certain constraint validations are skipped when adding, updating, or removing [**directory objects**](#gt_407dbc2c-3140-4e31-9085-0087e2d3bab2) on the opened connection. The skipped constraint validations are documented in the applicable constraint sections in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a).

The global variable [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219) for a DC has an associated field dc.ldapConnections, which maintains the DC's LDAPConnections state.

## LinkStamp

1. procedure LinkStamp(
2. o: DSName;
3. att: ATTRTYP;
4. val: attribute value): LinkValueStamp

The LinkStamp procedure returns the [LinkValueStamp](#Section_6a9517897afa47dda96c83fc0e30aa3d) associated with the last [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) to add or remove value *val* from the [**forward link attribute**](#gt_ca910b1e-dfb2-4a06-94a8-425013020fb9) att of [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) *o*. If *val* was last updated when the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) functional level was DS\_BEHAVIOR\_WIN2000 (see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.4.4), no LinkValueStamp is associated with *val*, and LinkStamp returns null.

## LinkValueStamp

LinkValueStamp is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that denotes information about the last [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) to a [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238) of an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). It is a tuple that consists of all the fields in [AttributeStamp](#Section_2973bb80c8ed450da98159639e09820b), plus the following additional fields:

* **timeCreated**: The date and time at which the first [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) was made.
* **timeDeleted**: The date and time at which the last [**replicated update**](#gt_2a923099-db0a-4932-af28-4354601e85c4) was made that deleted the value, or 0 if the value is not currently deleted.
* **timeExpired**: The date and time at which the value must be removed from the state of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**Comparisons**

Values of LinkValueStamp are partially ordered. Let d be the result of x.dwVersion - y.dwVersion, cast as a 32-bit integer. Then given two [**stamps**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) *x* and *y*, *x* is said to be greater than *y* if any of the following is true:

* *x* is not null and *y* is null
* x.timeCreated > y.timeCreated
* x.timeCreated = y.timeCreated and *d* > 0
* x.timeCreated = y.timeCreated and *d* = 0 and x.timeChanged > y.timeChanged
* x.timeCreated = y.timeCreated and *d* = 0 and x.timeChanged = y.timeChanged and x.uuidOriginating > y.uuidOriginating

**Conversions**

A value *x* of type LinkValueStamp can be converted to and from its wire format *y* of type [PROPERTY\_META\_DATA\_EXT](#Section_aef7ebdec305422495fd585c86b19c38) by associating the values of fields in *x* with the values of the like-named fields in *y* and y.MetaData.

**Note**  The value of timeDeleted does not appear in the wire format. On the wire, the PROPERTY\_META\_DATA\_EXT value always appears as a value of the MetaData field of a [REPLVALINF\_V1](#Section_22946fbf170e4ab482c7dabdfd97bf5a) or [REPLVALINF\_V3](#Section_9c15369bb7d2437ab73d66a92c367795) structure. Given value *x* of type LinkValueStamp and value *z* of type REPLVALINF\_V1 or REPLVALINF\_V3, z.fIsPresent is TRUE if x.timeDeleted is 0 and FALSE if x.timeDeleted is nonzero.

**Note** The value of timeExpired does not appear in the REPLVALINF\_V1 wire format. In cases where the REPLVALINF\_V1 wire format is used, the value of timeExpired is not returned to the caller.

## LinkValueStampCompare

1. procedure LinkValueStampCompare(
2. LinkValueStamp stamp1,
3. LinkValueStamp stamp2): integer

*Informative summary of behavior*: The LinkValueStampCompare procedure compares two [LinkValueStamps](#Section_6a9517897afa47dda96c83fc0e30aa3d), *stamp1* and *stamp2*. If *stamp1* > *stamp2* then the procedure returns an integer with a value greater than 0. If *stamp1* = *stamp2* then the procedure returns 0. If *stamp1* < *stamp2* then the procedure returns an integer value less than 0.

1. d: integer
2. d := 0
3. if stamp1.dwVersion ≠ 0 and stamp2.dwVersion = 0 then
4. d := 1
5. else if stamp1.dwVersion = 0 and stamp2.dwVersion ≠ 0 then
6. d := -1
7. endif
8. if d = 0 then
9. if stamp1.timeCreated > stamp2.timeCreated then
10. d := 1
11. else if stamp1.timeCreated < stamp2.timeCreated then
12. d := -1
13. endif
14. endif
15. if d = 0 then
16. /\* The value of d will be the re result of stamp1.dwVersion -
17. \* stamp2.dwVersion, cast as a 32-bit integer. For example, if
18. \* stamp1.dwVersion is 1 and stamp2.dwVersion is 3, d is -2. If
19. \* stamp1.dwVersion is 5 and stamp2.dwVersion is 0xFFFFFFFA,
20. \* d is 11.
21. \*/
22. d := stamp1.dwVersion - stamp2.dwVersion
23. endif
24. if d = 0 then
25. if stamp1.timeChanged > stamp2.timeChanged then
26. d := 1
27. else if stamp1.timeChanged < stamp2.timeChanged then
28. d := -1
29. endif
30. endif
31. if d = 0 then
32. if stamp1.uuidOriginating > stamp2.uuidOriginating then
33. d := 1
34. else if stamp1.uuidOriginating < stamp2.uuidOriginating then
35. d := -1
36. endif
37. endif
38. return d

## LocalAttidFromRemoteAttid

1. procedure LocalAttidFromRemoteAttid(
2. remotePT: PrefixTable,
3. remoteAttid : ATTRTYP) : ATTRTYP

*Informative summary of behavior*: The LocalAttidFromRemoteAttid procedure converts the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) ID *remoteAttid* based on the [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) *remotePT* to an attribute ID based on [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).prefixTable.

1. oid : OID
2. oid := OidFromAttid(remotePT, remoteAttid)
3. return MakeAttid(dc.prefixTable, oid)

## LONG

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a 32-bit, signed integer, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.27.

## LONGLONG

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a 64-bit, signed integer, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.28.

## LPWSTR

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a pointer to a string of double-byte [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) characters, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.36.

## MakeAttid

1. procedure MakeAttid(var t: PrefixTable, o: OID) : ATTRTYP

The MakeAttid procedure translates an abstract [OID](#Section_339504853a964b668a28a3a33e80302b) *o* to a concrete [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983), using the [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) specified by *t*. This procedure can mutate the supplied prefix table. See section [5.16.4](#Section_6f53317f226348ee86c14580bf97232c) for the specification of this procedure.

## MakeProxyValue

1. procedure MakeProxyValue(
2. dnPart: DSName,
3. proxyType: DWORD,
4. proxyEpoch: DWORD): DNBinary

The MakeProxyValue procedure constructs and returns a value in the proxiedObjectName format (section [5.157](#Section_0da1cf6f630a4fd5939f757b950d63b0)) from the provided parts. Let *d* be the returned [DNBinary](#Section_b8def843aed54dcb81d99691956e6f1a). *d.dn* equals *dnPart* and *d.binary* is constructed from *proxyType* and *proxyEpoch*.

## MasterReplicaExists

1. procedure MasterReplicaExists(nc : DSName) : boolean

The MasterReplicaExists procedure returns true only if the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) with root *nc* is a [**writable NC replica**](#gt_51db485c-dcf6-4845-99b3-2df414ef0aa9).

1. If not ObjExists(nc) then
2. return false
3. endif
4. return nc in DSAObj()!msDS-hasMasterNCs

## MD5\_CTX

MD5\_CTX is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) defined in [[RFC1321]](https://go.microsoft.com/fwlink/?LinkId=90275).

## MD5Final

1. procedure MD5Final(var context: MD5\_CTX)

The MD5Final procedure performs the MD5Final algorithm, as specified in [[RFC1321]](https://go.microsoft.com/fwlink/?LinkId=90275).

## MD5Init

1. procedure MD5Init(var context: MD5\_CTX)

The MD5Init procedure performs the MD5Init algorithm, as specified in [[RFC1321]](https://go.microsoft.com/fwlink/?LinkId=90275).

## MD5Update

1. procedure MD5Update(
2. var context: MD5\_CTX,
3. input: sequence of BYTE,
4. inputLen: integer)

The MD5Update procedure performs the MD5Update algorithm, as specified in [[RFC1321]](https://go.microsoft.com/fwlink/?LinkId=90275).

## MergeUTD

1. procedure MergeUTD(
2. utd1: UPTODATE\_VECTOR\_V1\_EXT,
3. utd2: UPTODATE\_VECTOR\_V1\_EXT): UPTODATE\_VECTOR\_V1\_EXT

*Informative summary of behavior*: The client does not want to include [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the inconsistency-detection process that have not yet replicated. To meet this goal, it uses the MergeUTD procedure to compute an [UPTODATE\_VECTOR\_V1\_EXT](#Section_462b424ab50a4c4aa81f48d0f4cf40fe) that has minimal pairwise values for each uuidDsa.

MergeUTD is specified by the following normative semantics:

For every uuidDsa that is in both *utd1* and *utd2*, add the uuidDsa to the returned UPTODATE\_VECTOR\_V1\_EXT with a corresponding [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) value such that the USN is the smaller of the USNs corresponding to the uuidDsa in *utd1* and *utd2*.

## MTX\_ADDR

The MTX\_ADDR structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the network name of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

1. typedef struct {
2. [range(1,256)] unsigned long mtx\_namelen;
3. [size\_is(mtx\_namelen)] char mtx\_name[];
4. } MTX\_ADDR;

**mtx\_namelen:**  A 32-bit, unsigned integer that specifies the number of bytes in **mtx\_name**, including a terminating null character.

**mtx\_name:**  The UTF-8 encoding of a [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912).

The following table shows an alternative representation of this structure.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| mtx\_namelen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| mtx\_name (variable length) ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

## NCType Bits

Bit flags, presented in little-endian byte order, describing **NCType**.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | X | X | F P | G P | S P | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**X**: Unused. MUST be zero and ignored.

**SP (NCT\_SPECIAL\_SECRET\_PROCESSING, 0x00000001)**: The [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) requests special secret processing while replicating [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

**GP (NCT\_GC\_PARTIAL, 0x00000002)**: Objects in the NC replica can only have [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that are specified in the [**GC partial attribute set**](#gt_88536a16-ced1-4fbb-8bf4-8e4d994562af).

**FP (NCT\_FILTERED\_ATTRIBUTE\_SET, 0x00000004)**: Objects in the NC replica do not have attributes defined in the [**filtered attribute set**](#gt_1bbc9ed8-f11c-4be6-8a41-1f396785602d).

## NetworkAddress

NetworkAddress is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) for the transport-specific address of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) represented as a string. For the SMTP transport, the address is an SMTP address (as specified in [[RFC2821]](https://go.microsoft.com/fwlink/?LinkId=90384) and [[MS-SRPL]](%5bMS-SRPL%5d.pdf#Section_ec69eea50d5e428ab5bc66732aaeb866)). For the [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a) in [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024), the NetworkAddress is a fully qualified DNS name corresponding to the DC.

For the RPC transport in [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab), the NetworkAddress is a UTF-8 string in the following format:

* <DC-name>:<DC-identifier>

In the preceding format:

* <DC-name> is an IP address in the UTF-8 format, a fully qualified DNS name, or a NetBIOS name.
* <DC-identifier> is either a [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) or an integer. The GUID corresponds to the [**objectGUID**](#gt_ad613dff-e9c4-4cb6-ad6b-0ce52038ceb5) [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of the DC's nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). The integer is the **ldapPort** attribute of the DC's nTDSDSA object.
* The colon (":") is the literal separator between the DC-name and the DC-identifier.

A NetworkAddress is stored as an mtx\_name within an [MTX\_ADDR](#Section_107b7c0e0f0d4fe2823214ec3b78f40d) structure or in a location that is pointed to by the **cbpszInstanceOffset** member of the [DSA\_RPC\_INST](#Section_88a396196dbe4ba184355966c1a490a7) structure, which in turn is stored within a [REPS\_TO](#Section_b422aa877d074527b070c5d719696c43) structure. The concrete representation of NetworkAddress in these concrete structures is the same as the abstract representation described above.

## NewPrefixTable

1. procedure NewPrefixTable() : PrefixTable

The NewPrefixTable procedure creates a new [PrefixTable](#Section_2789d96b50e8444d82d6523831556d76) that contains a set of default prefixes. See section [5.16.4](#Section_6f53317f226348ee86c14580bf97232c) for the specification of this procedure.

## Nt4ReplicationState

Nt4ReplicationState is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) for the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state for Windows NT 4.0 BDCs. It is a tuple that contains the following fields:

* **SamNT4ReplicationUSN**: A [USN](#Section_1be1e991a2db4f9199538eab69f60e64) that records the replication [**update sequence number**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) for SAM database [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) that are relevant to the Windows NT 4.0 replication protocol. Relevant updates are described in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.7.2.2.
* **SamCreationTime**: A [FILETIME](#Section_70ee934bc9b944498aa36dfe9cef3eff) at which the Windows NT 4.0 replication update sequence number for the SAM database was set to 1.
* **BuiltinNT4ReplicationUSN**: A USN that records the replication update sequence number for built-in database updates that are relevant to the Windows NT 4.0 replication protocol. The built-in database contains the [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) for [**built-in principals**](#gt_232336e6-3c6a-4c6b-8c35-52bbcf3c090a).
* **BuiltinCreationTime**: A FILETIME at which the replication update sequence number for the built-in database was set to 1.

The global variable [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219) for a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) has an associated field dc.nt4ReplicationState. When a DC owns the [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f), this field contains its Nt4ReplicationState. [MS-ADTS] section 3.1.1.7.1.1 describes how the components of dc.nt4ReplicationState are maintained and used to support replication via the Windows NT 4.0 replication mechanism. As an implementation-specific behavior, other DCs might maintain the dc.nt4ReplicationState field as well.

## NT4SID

The NT4SID structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d).

1. typedef struct {
2. char Data[28];
3. } NT4SID;

**Data:**  Bytes that make up a SID structure, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2, in little-endian byte order.

## NTSAPI\_CLIENT\_GUID

NTDSAPI\_CLIENT\_GUID is a value of type [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) that is defined as {e24d201a-4fd6-11d1-a3da-0000f875ae0d}.

## NTDSTRANSPORT\_OPT Values

The valid system flags used on [**directory objects**](#gt_407dbc2c-3140-4e31-9085-0087e2d3bab2) are specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.1.2.2.3.1.

## NULLGUID

NULLGUID is a value of type [GUID](#Section_5e740f50e6a048c9bca800072e85d963) that is entirely zero, that is, {00000000-0000-0000-0000-000000000000}. This is a constant representation of the [**NULL GUID**](#gt_ba500a5b-8c29-467c-a335-0980c8b11304) value.

## ObjExists

1. procedure ObjExists(dsName: DSName): boolean

The ObjExists procedure returns true if *dsName* identifies an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in some [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) hosted by the server.

1. rt: DSName
2. rt:= select one v from all where v = dsName
3. if (rt = null) then
4. return false
5. else
6. return true
7. endif

## OID

[**OID**](#gt_aaaf2f1a-0b0a-487e-a0f0-c3510a6091b2) is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) for representing values of type String(Object-Identifier). Values of this type are a dotted decimal *unicodestring* (section [3.4.3](#Section_fbe9988847824858b5f25b521a44d836)), for example, "1.2.840.113556.1.4.159".

## OID\_t

The OID\_t structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for an [OID](#Section_339504853a964b668a28a3a33e80302b) or a prefix of an OID; it is a component of type [PrefixTableEntry](#Section_d26d36cd10c44b27a84e98336abf357a).

1. typedef struct {
2. [range(0,10000)] unsigned int length;
3. [size\_is(length)] BYTE\* elements;
4. } OID\_t;

**length:**  The size, in bytes, of the elements array.

**elements:**  An array of bytes that constitute an OID or a prefix of an OID.

## OidFromAttid

1. procedure OidFromAttid(t: PrefixTable, attr: ATTRTYP) : OID

The OidFromAttid procedure translates a concrete [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) *attr* to an abstract [OID](#Section_339504853a964b668a28a3a33e80302b), using the [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) specified by *t*. See section [5.16.4](#Section_6f53317f226348ee86c14580bf97232c) for the specification of this procedure.

## parent

parent is an abstract [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that is present on every [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca), as specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.1.3. This attribute is part of the state model but is not exposed in the [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093).

## PARTIAL\_ATTR\_VECTOR\_V1\_EXT

The PARTIAL\_ATTR\_VECTOR\_V1\_EXT structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a set of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) to be replicated to a given partial [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac).

1. typedef struct {
2. DWORD dwVersion;
3. DWORD dwReserved1;
4. [range(1,1048576)] DWORD cAttrs;
5. [size\_is(cAttrs)] ATTRTYP rgPartialAttr[];
6. } PARTIAL\_ATTR\_VECTOR\_V1\_EXT;

**dwVersion:**  The version of this structure; MUST be 1.

**dwReserved1:**  Unused. MUST be 0 and ignored.

**cAttrs:**  The number of attributes in the rgPartialAttr array.

**rgPartialAttr:**  The attributes in the set.

## partialAttributeSet

The abstract, nonreplicated, single-valued [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) partialAttributeSet is an optional attribute on the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of every partial [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac).

The [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) set of [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) simplifies the specification of methods that read and write the attribute partialAttributeSet. Reading the attribute partialAttributeSet returns a single value, which is of type set of ATTRTYP. Each element in the set is an attribute that is in the subset of attributes replicated to the partial replica.

## PartialGCReplicaExists

1. procedure PartialGCReplicaExists(nc : DSName) : boolean

The PartialGCReplicaExists procedure returns true if the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) with root *nc* is a [**partial NC replica**](#gt_2d142c30-79c2-47f7-81d0-6ae878c5db2c).

1. if not ObjExists(nc) then
2. return false
3. endif
4. return nc in DSAObj()!hasPartialReplicaNCs

## PAS\_DATA

**PAS\_DATA** is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a list of [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) in a [**partial attribute set**](#gt_2b3cc270-8a21-4402-bb8b-9bebac24bdaa).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| version | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| size | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| flag | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| pas (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**version (4 bytes):** The version of the structure; MUST be 1.

**size (4 bytes):** The size of the entire structure.

**flag (4 bytes):** Unused. MUST be 0 and ignored.

**pas (variable):** A [PARTIAL\_ATTR\_VECTOR\_V1\_EXT](#Section_1d5c1b34daa44761a8b5d3c0146a0e30) structure, which specifies the additional attributes being requested as part of a PAS [**cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16).

## PdcChangeLog

PdcChangeLog is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) for a sequence of [CHANGELOG\_ENTRY](#Section_0861744b5ee0428fb11aa25092636b64) [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). The change log is used to support [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) of certain [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) changes to Windows NT 4.0 BDCs, and is specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.7.

The global variable [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219) for a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) has an associated field dc.pdcChangeLog. When a DC owns the [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) [**FSMO role**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f), this field contains its change log. As an implementation-specific behavior, other DCs might maintain the dc.pdcChangeLog field as well.

## PerformAddOperation

1. procedure PerformAddOperation(
2. data: ENTINF,
3. var newObjectName: DSName,
4. prefixTable: PrefixTable,
5. boolean: fAsOriginating): integer

The PerformAddOperation procedure performs an add operation with the given [ENTINF](#Section_6d69822eadb649778553c2d529c17e5b) to create a new [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) in the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9). For more details, see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.5.2.

The resulting object has the [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) *data*.pName.

For each [ATTR](#Section_a2db41e278034d3ca4990fee92b1c149) **attr** in *data*.AttrBlock, let *attribute* be the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) returned by [LocalAttidFromRemoteAttid](#Section_5e30bd01b1dd4019b4061e9c2472f359)(*prefixTable*, attr.attrType). Then the object created by PerformAddOperation has an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) whose ATTRTYP is *attribute* and that has the values attr.AttrVal.pAVal[0... attr.AttrVal.valCount].

If data.ulFlags ∩ {ENTINF\_DYNAMIC\_OBJECT} = {ENTINF\_DYNAMIC\_OBJECT}, the resulting object is created as a [**dynamic object**](#gt_ea6b6f3f-6bed-4622-aaca-fd7df28badb9).

If *fAsOriginating* is true, then add the object as an originating [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493). See [MS-ADTS] section 3.1.1.1.9.

If the add operation succeeds, the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of the created object is returned in *newObjectName* and the procedure returns 0. If the add operation fails, the procedure returns a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b).

## PerformAddOperationAsSystem

1. procedure PerformAddOperationAsSystem(
2. data: ENTINF,
3. prefixTable: PrefixTable,
4. var newObjectName: DSNAME): integer

The PerformAddOperationAsSystem procedure is identical to [PerformAddOperation](#Section_e61c9ac3cd7c47d3ab1b0c8b61dc4869), except that the add operation is performed as the system. When an operation is performed as the system, all access checks are bypassed and [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093) constraints are not enforced.

## PrefixTable

PrefixTable is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) for a [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe). See section [5.16.4](#Section_6f53317f226348ee86c14580bf97232c) for the specification of this type.

## PrefixTableEntry

The PrefixTableEntry structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for mapping a range of [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values to and from [OID](#Section_339504853a964b668a28a3a33e80302b)s. It is a component of the type [SCHEMA\_PREFIX\_TABLE](#Section_9b371267e8b84c69997902dae02e5e38).

1. typedef struct {
2. unsigned long ndx;
3. OID\_t prefix;
4. } PrefixTableEntry;

**ndx:**  The index assigned to the prefix.

**prefix:**  An OID or a prefix of an OID.

## PROPERTY\_META\_DATA\_EXT

The PROPERTY\_META\_DATA\_EXT structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) of the last [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) to an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).

1. typedef struct {
2. DWORD dwVersion;
3. DSTIME timeChanged;
4. UUID uuidDsaOriginating;
5. USN usnOriginating;
6. } PROPERTY\_META\_DATA\_EXT;

**dwVersion:**  The version of the attribute values, starting at 1 and increasing by one with each originating update.

**timeChanged:**  The time at which the originating update was performed.

**uuidDsaOriginating:**  The invocationId of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that performed the originating update.

**usnOriginating:**  The [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) of the DC assigned to the originating update.

## PROPERTY\_META\_DATA\_EXT\_VECTOR

The PROPERTY\_META\_DATA\_EXT\_VECTOR structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a sequence of [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) [**stamps**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00).

1. typedef struct {
2. [range(0,1048576)] DWORD cNumProps;
3. [size\_is(cNumProps)] PROPERTY\_META\_DATA\_EXT rgMetaData[];
4. } PROPERTY\_META\_DATA\_EXT\_VECTOR;

**cNumProps:**  The number of items in the **rgMetaData** array.

**rgMetaData:**  An array of attribute stamps.

## proxiedObjectName Value Format

Values of the proxiedObjectName [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) are of [DNBinary](#Section_b8def843aed54dcb81d99691956e6f1a) type. The binary portion is composed of two [DWORD](#Section_60c3f5f194924d1083c89a155e162ef3)s, which are stored in big-endian format. The first DWORD contains the "proxy type" value. The following table lists the valid values for the first DWORD.

| Symbolic name | Value | Meaning |
| --- | --- | --- |
| PROXY\_TYPE\_MOVED\_OBJECT | 0x0 | An [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) that was cross-[**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) moved. |
| PROXY\_TYPE\_PROXY | 0x1 | Used by the reference [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) task. For more details, see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.6.2. |

The second DWORD is the "proxy epoch" value, which is a DWORD counter value that is used by the cross-NC move operation.

## RDN

[**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) for representing the relative distinguished name (RDN) (as specified in [[RFC2253]](https://go.microsoft.com/fwlink/?LinkId=90327)).

## rdnType

rdnType is an abstract [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) present on every [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). The rdnType of an object is the [**RDN**](#gt_22198321-b40b-4c24-b8a2-29e44d9d92b9) attribute of the object, that is, an [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) that identifies the attributeSchema object of the RDN attribute. rdnType is not represented in the [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093) and does not replicate in the normal way.

On an originating Add, the new object's rdnType is derived from the most specific structural [**object class**](#gt_a224e395-3fea-48bd-b141-3dd9bee2136a) of the new object.

On a replicated Add, rdnType is derived as follows:

* If the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) functional level is less than DS\_BEHAVIOR\_WIN2003, rdnType is derived from the [**objectClass**](#gt_4191a0f6-e528-4927-bf0e-7a9981e014c8) of the object, which replicates.
* If the forest functional level is DS\_BEHAVIOR\_WIN2003 or greater, rdnType is derived from the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the object, which replicates.

## RecycleObj

1. procedure RecycleObj(o: DSName)

The RecycleObj procedure transforms, as described in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.5.5, the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) whose DSName is *o* into a [**recycled-object**](#gt_156b927d-f1ce-4629-993f-18f0cd5e1e12). All appropriate [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) (possibly including distinguishedName) are changed or removed from the object to conform to the invariants of [MS-ADTS] section 3.1.1.5.5. Any changes that need to be made to the object are performed as an [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20), except for changes required to remove linked attribute values, which are simply removed from the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9). Attributes and values that already conform to the invariants are not changed. See [MS-ADTS] section 3.1.1.1.9 for more details on originating updates.

## RemoveObj

1. procedure RemoveObj(o: DSName,treeDeletion: boolean): ULONG

The RemoveObj procedure performs a delete operation on the [object](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) whose [DSName](#Section_a0d5477a522946b9890a54b924d487d1) is *o*. If the value of parameter *treeDeletion* is true, then the tree-delete variation of the operation is performed. As described in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.5.5, the delete operation transforms the targeted object into a [deleted-object](#gt_d9c9e99f-74f1-483e-bcb1-310e75ff1344) or a [tombstone](#gt_9d8e0963-13fa-4e19-a97f-7ce6bc90d20f), depending on the state of the [Recycle Bin](#gt_54624800-58f4-45e9-90bf-c9b52dcf98f3) [optional feature](#gt_785b66f1-22b3-450f-97aa-a24a39d04d47). The tree-delete operation performs, as described in [MS-ADTS] section 3.1.1.5.5.7.3, a delete operation on all objects in the subtree rooted at the [target object](#gt_62c95f88-0024-410c-b008-b637d04803ad). All appropriate [attributes](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) (possibly including distinguishedName) are changed or removed from the deleted objects to conform to the invariants of [MS-ADTS] section 3.1.1.5.5. Any changes that need to be made to the objects are performed as an [originating update](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20), except for changes required to remove linked attribute values, which are simply removed from the [directory](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9). Attributes and values that already conform to the invariants are not changed. See [MS-ADTS] section 3.1.1.1.9 for more details on originating updates. If the delete operation succeeds, 0 is returned. Otherwise, this procedure returns an error code, as specified in [MS-ADTS] section 3.1.1.5.5, that indicates the reason for the failure.

## REPLENTINFLIST

The REPLENTINFLIST structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) to one or more [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) of a given [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca).

1. typedef struct REPLENTINFLIST {
2. struct REPLENTINFLIST\* pNextEntInf;
3. ENTINF Entinf;
4. BOOL fIsNCPrefix;
5. UUID\* pParentGuid;
6. PROPERTY\_META\_DATA\_EXT\_VECTOR\* pMetaDataExt;
7. } REPLENTINFLIST;

**pNextEntInf:**  The next REPLENTINFLIST in the sequence, or null.

**Entinf:**  The object and its updated attributes.

**fIsNCPrefix:**  TRUE only if the object is an [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root.

**pParentGuid:**  The value of the objectGUID attribute of the parent of the object, or null if not known or not specified.

**pMetaDataExt:**  The [**stamps**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) for the attributes specified in Entinf.AttrBlock. Entinf.AttrBlock and pMetaDataExt.rgMetaData are parallel arrays. For a given integer *i* in [0 .. Entinf.AttrBlock.attrCount], the stamp for the attribute described by Entinf.AttrBlock.pAttr^[*i*] is pMetaDataExt^.rgMetaData[*i*].

## ReplicatedAttributes

1. procedure ReplicatedAttributes(): Set of ATTRTYP

The ReplicatedAttributes procedure returns the set of [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) of all [**attributes**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) in the [**schema**](#gt_fd49ea36-576c-4417-93bd-d1ac63e71093) excluding [**nonreplicated attributes**](#gt_6c9b51bd-519b-4f20-97ae-baaf9675f2d7). See [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.2.3 for more details on nonreplicated attributes.

## ReplicationQueue

ReplicationQueue is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) for queued pending [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) operations. It is a sequence of tuples, one tuple for each queued replication operation that is pending. Each tuple contains the following fields:

* **TimeEnqueued**: A [FILETIME](#Section_70ee934bc9b944498aa36dfe9cef3eff) that contains the time when the operation was enqueued.
* **SerialNumber**: A [ULONG](#Section_20419b45c61d47ccb4fc0b2ab66934cc) that contains a unique identifier associated with the operation.
* **Priority**: A ULONG that contains the priority of the operation. Tasks with a higher priority value are executed first. The priority is calculated by the server based on the type of operation and its parameters.
* **OperationType**: An integer that indicates the type of operation, as defined in [DS\_REPL\_OP\_TYPE](#Section_bf047cfe32bd43f693d3b67b05eaac66).
* **Options**: A [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) that contains options associated with the replication operation.
* **NamingContext**: A *unicodestring* (section [3.4.3](#Section_fbe9988847824858b5f25b521a44d836)) that contains the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of the [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) associated with the operation.
* **DsaDN**: A *unicodestring* (section 3.4.3) that contains the [**DN**](#gt_1175dd11-9368-41d5-98ed-d585f268ad4b) of the nTDSDSA [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) associated with the operation.
* **DsaAddress**: A *unicodestring* (section 3.4.3) that contains the network address of the DC associated with the operation.
* **UUIDNC**: A [GUID](#Section_5e740f50e6a048c9bca800072e85d963) that contains the objectGUID of the NC root of the NC replica associated with the operation.
* **UUIDDsa**: A GUID that contains the [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the DC associated with the operation.

The global variable [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219) for a DC has an associated field dc.replicationQueue, which maintains the DC's ReplicationQueue state.

## REPLTIMES

The REPLTIMES structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for times at which periodic [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) occurs.

1. typedef struct {
2. UCHAR rgTimes[84];
3. } REPLTIMES;

**rgTimes:**  A byte array of length 84 that is used to set periodic replication times. Each bit in this byte array represents a 15-minute period for which replication can be scheduled within a one-week period. The replication schedule begins on Sunday 12:00:00 AM UTC. Each byte in the array represents a two-hour period of a week in ascending order, starting Sunday 12:00:00 AM UTC. The most significant bit of a byte represents the earliest 15-minute period in the two-hour period, and the rest of the bits in the byte represent their respective 15-minute periods in this order.

The following diagram shows an alternative representation of this structure.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| rgTimes... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ...rgTimes | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

## replUpToDateVector, ReplUpToDateVector

The [**nonreplicated attribute**](#gt_6c9b51bd-519b-4f20-97ae-baaf9675f2d7) replUpToDateVector is an optional [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) on the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root of every [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

The [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) ReplUpToDateVector simplifies the specification of methods that read and write the replUpToDateVector attribute. Reading the replUpToDateVector attribute produces one or more ReplUpToDateVector values.

The ReplUpToDateVector type is a tuple with the following fields:

* **uuidDsa**: The [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that assigned usnHighPropUpdate.
* **usnHighPropUpdate**: A [USN](#Section_1be1e991a2db4f9199538eab69f60e64) at which an [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) was applied on the DC identified by uuidDsa.
* **timeLastSyncSuccess**: The time at which the last successful [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) occurred from the DC identified by uuidDsa; for [**replication latency**](#gt_2352e9b3-ae08-4b5f-8858-bbca4ff4dd97) reporting only.

Given an NC replica r, if c is an element of r!replUpToDateVector, then all updates made by c.uuidDsa with [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) <= c.usnHighPropUpdate have been applied to r.

## REPLVALINF\_V1

The REPLVALINF\_V1 structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the identity and [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) of a [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238).

1. typedef struct {
2. DSNAME\* pObject;
3. ATTRTYP attrTyp;
4. ATTRVAL Aval;
5. BOOL fIsPresent;
6. VALUE\_META\_DATA\_EXT\_V1 MetaData;
7. } REPLVALINF\_V1;

**pObject:**  Identifies the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) with the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that contains the link value.

**attrTyp:**  An attribute that contains the link value.

**Aval:**  The link value.

**fIsPresent:**  FALSE if and only if the link value has been removed from the attribute.

**MetaData:**  The stamp associated with the link value.

## REPLVALINF\_V3

The REPLVALINF\_V3 structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the identity and [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) of a [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238). This structure contains all the same elements as a [**REPLVALINF\_V1**](#Section_22946fbf170e4ab482c7dabdfd97bf5a) structure except that the data type of MetaData is changed from [**VALUE\_META\_DATA\_EXT\_V1**](#Section_7530cf2ea2ad4716a5708383f8b1846f) to [**VALUE\_META\_DATA\_EXT\_V3**](#Section_eab72899a828427d83849a51ffdb77e1). Because **VALUE\_META\_DATA\_EXT\_V3** is a superset of **VALUE\_META\_DATA\_EXT\_V1**, this structure is a superset of the **REPLVALINF\_V1** structure.

1. typedef struct {
2. DSNAME\* pObject;
3. ATTRTYP attrTyp;
4. ATTRVAL Aval;
5. BOOL fIsPresent;
6. VALUE\_META\_DATA\_EXT\_V3 MetaData;
7. } REPLVALINF\_V3;

**pObject**: Identifies the [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) with the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) that contains the link value.

**attrTyp**: An attribute that contains the link value.

**Aval**: The link value.

**fIsPresent**: FALSE if and only if the link value has been removed from the attribute.

**MetaData**: The stamp associated with the link value.

## REPLVALINF\_NATIVE

The REPLVALINF\_NATIVE structure is an alias for the [**REPLVALINF\_V3**](#Section_9c15369bb7d2437ab73d66a92c367795) data structure.

## REPS\_FROM

The nonreplicated, multivalued [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) repsFrom is an optional attribute on the root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of every [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). It is stored with the structure of the REPS\_FROM [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d), which is represented by the following diagram.

**Note**  In the following field descriptions, the source [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) refers to the DC identified by the uuidDsaObj.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| dwVersion | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| dwReserved0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cb | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cConsecutiveFailures | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| timeLastSuccess | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| timeLastAttempt | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ulResultLastAttempt | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cbOtherDraOffset | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cbOtherDra | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ulReplicaFlags | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| rtSchedule (84 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| dwReserved1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| usnVec (24 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| uuidDsaObj (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| uuidInvocId (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| uuidTransportObj (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| dwReserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cbPasDataOffset | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| data (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**dwVersion (4 bytes):** The version of this structure. The value must be 1 or 2.[<49>](#Appendix_A_49" \o "Product behavior note 49)

**dwReserved0 (4 bytes):** Unused. MUST be 0 and ignored.

**cb (4 bytes):** The total number of bytes in the REPS\_FROM structure.

**cConsecutiveFailures (4 bytes):** An unsigned long that contains the number of consecutive failures that have occurred while replicating from the source DC.

**timeLastSuccess (8 bytes):** A [DSTIME](#Section_a72a16b973e441caa5c1afc5fc54e175) that contains the time of the last successful [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) [**cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16) with the source DC.

**timeLastAttempt (8 bytes):** A DSTIME that contains the time of the last replication attempt with the source DC.

**ulResultLastAttempt (4 bytes):** A Win32 error code, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.2, that represents the result of the last replication attempt with the source DC.

**cbOtherDraOffset (4 bytes):** The offset from the start of the structure to a location in the data field, specifying the start of a structure that contains a [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) for the source DC. If **dwVersion** is 1, it is an [MTX\_ADDR](#Section_107b7c0e0f0d4fe2823214ec3b78f40d) structure. If **dwVersion** is 2, it is a [DSA\_RPC\_INST](#Section_88a396196dbe4ba184355966c1a490a7) structure.

**cbOtherDra (4 bytes):** The size of the structure pointed to by **cbOtherDraOffset**.

**ulReplicaFlags (4 bytes):** A ULONG. This field contains a set of [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) that are applicable when replicating from the source DC.

**rtSchedule (84 bytes):** A [REPLTIMES](#Section_42d7d8e8794e427998028b5916e8b099) structure. If periodic replication is enabled (ulReplicaFlags contains DRS\_PER\_SYNC), this field identifies the 15-minute intervals within each week when a replication cycle is initiated with the source DC.

**dwReserved1 (4 bytes):** Unused. MUST be 0 and ignored.

**usnVec (24 bytes):** A [USN\_VECTOR](#Section_595d11b86ca74a61bd563e6a2b99b76b) structure. This holds 0 or the usnvecTo field from the response to the last IDL\_DRSGetNCChanges replication request sent to the source DC.

**uuidDsaObj (16 bytes):** A GUID that is the [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the source DC.

**uuidInvocId (16 bytes):** A GUID that contains the [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the source DC.

**uuidTransportObj (16 bytes):** A GUID that contains the objectGUID of the interSiteTransport object that corresponds to the transport used for communication with the source DC.

**dwReserved (4 bytes):** Unused. MUST be 0 and ignored.

**cbPasDataOffset (4 bytes):** The offset from the start of the structure to a location in the data field, specifying the start of a PAS\_DATA structure.

**data (variable):** The storage for the rest of the structure. The structures pointed to by **cbOtherDraOffset** and **cbPasDataOffset** are packed into this field and can be referenced using the offsets.

## REPS\_TO

The nonreplicated, multivalued [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) repsTo is an optional attribute on the root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of every [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). It is stored with the structure of the REPS\_TO [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d), which is represented by the following diagram.

This structure is used for both repsTo values and repsFrom values. Many of the fields are unused in repsTo values, and some of the field names are misleading (for example, **timeLastSuccess**).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| dwVersion | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| dwReserved0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cb | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cConsecutiveFailures | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| timeLastSuccess | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| timeLastAttempt | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ulResultLastAttempt | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cbOtherDraOffset | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cbOtherDra | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ulReplicaFlags | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| rtSchedule (84 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| dwReserved1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| usnVec (24 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| uuidDsaObj (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| uuidInvocId (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| uuidTransportObj (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| dwReserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| cbPasDataOffset | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| data (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**dwVersion (4 bytes):** The version of this structure. The value must be 1 or 2.[<50>](#Appendix_A_50" \o "Product behavior note 50)

**dwReserved0 (4 bytes):** Unused. MUST be 0 and ignored.

**cb (4 bytes):** The total number of bytes in the REPS\_TO structure.

**cConsecutiveFailures (4 bytes):** An unsigned long that contains the number of unsuccessful consecutive attempts to send a [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) notification to the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) identified by uuidDsaObj.

**timeLastSuccess (8 bytes):** A [DSTIME](#Section_a72a16b973e441caa5c1afc5fc54e175) structure that contains the time when the last successful replication notification to the DC identified by uuidDsaObj was sent, or 0 if no replication notification has been sent successfully.

**timeLastAttempt (8 bytes):** A DSTIME structure that contains the last time when an attempt was made to send a replication notification to the DC identified by uuidDsaObj, or 0 if no attempt has been made.

**ulResultLastAttempt (4 bytes):** An unsigned long that contains the result of the last attempt to send a replication notification to the DC identified by uuidDsaObj. It has a value of 0 if the last notification was sent successfully or a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) (as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.2) otherwise.

**cbOtherDraOffset (4 bytes):** The offset from the start of the structure to a location in the data field, specifying the start of a structure that contains a [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912). If **dwVersion** is 1, it is an [MTX\_ADDR](#Section_107b7c0e0f0d4fe2823214ec3b78f40d) structure. If **dwVersion** is 2, it is a [DSA\_RPC\_INST](#Section_88a396196dbe4ba184355966c1a490a7) structure.

**cbOtherDra (4 bytes):** The size of the structure pointed to by **cbOtherDraOffset**.

**ulReplicaFlags (4 bytes):** A ULONG. This set contains DRS\_WRIT\_REP (section [5.41](#Section_ac9c8a11cd464080acbf9faa86344030)) if this [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) is writable. This set never contains any other elements.

**rtSchedule (84 bytes):** A [REPLTIMES](#Section_42d7d8e8794e427998028b5916e8b099) structure. Not used.

**dwReserved1 (4 bytes):** Unused. MUST be 0 and ignored.

**usnVec (24 bytes):** A [USN\_VECTOR](#Section_595d11b86ca74a61bd563e6a2b99b76b) structure. Not used.

**uuidDsaObj (16 bytes):** A [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1). A [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) that identifies a DC.

**uuidInvocId (16 bytes):** A GUID. Not used.

**uuidTransportObj (16 bytes):** A GUID. Not used.

**dwReserved (4 bytes):** Unused. MUST be 0 and ignored.

**cbPasDataOffset (4 bytes):** Not used.

**data (variable):** The storage for the rest of the structure. The structure pointed to by **cbOtherDraOffset** is packed into this field and can be referenced using the offset.

## repsFrom, RepsFrom

The nonreplicated, multivalued [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) repsFrom is an optional attribute on the root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of every [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). It is stored with the structure [REPS\_FROM](#Section_f8e930ead84745858d58993e05f55e45).

The [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) RepsFrom simplifies the specification of methods that read and write the attribute repsFrom. Reading the attribute repsFrom produces one or more RepsFrom values using the conversions from REPS\_FROM specified below. Writing a RepsFrom value to the attribute repsFrom stores a REPS\_FROM using the reverse conversion.

The type RepsFrom is a tuple with the following fields:

**naDsa:** A [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) that corresponds to cbOtherDraOffset and cbOtherDra in REPS\_FROM. This is a NetworkAddress of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**uuidDsa:** A [GUID](#Section_5e740f50e6a048c9bca800072e85d963) that corresponds to uuidDsaObj in REPS\_FROM. This is the [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the DC.

**options:** A [ULONG](#Section_20419b45c61d47ccb4fc0b2ab66934cc) that corresponds to ulReplicaFlags in REPS\_FROM. This set contains one or more of the following values chosen from [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030):

* DRS\_WRIT\_REP: The [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) is a full (read/write) replica of the [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942).
* DRS\_INIT\_SYNC: The replica must be replicated from the DC identified by uuidDsa when the DC hosting this replica is started.
* DRS\_PER\_SYNC: Periodically replicate the NC replica from the DC identified by uuidDsa, as defined by the periodic [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) schedule.
* DRS\_MAIL\_REP: Replicate the NC replica from the DC identified by uuidDsa via SMTP (see [[MS-SRPL]](%5bMS-SRPL%5d.pdf#Section_ec69eea50d5e428ab5bc66732aaeb866)).
* DRS\_DISABLE\_AUTO\_SYNC: Disable notification-based replication of the NC replica from the DC identified by uuidDsa.
* DRS\_DISABLE\_PERIODIC\_SYNC: Disable periodic replication of the NC replica from the DC identified by uuidDsa.
* DRS\_USE\_COMPRESSION: Replication response messages sent along this communication path must be compressed.
* DRS\_TWOWAY\_SYNC: At the end of a replication [**cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16), replication must be triggered in the opposite direction.

The following additional values are preserved if they are present when reading **ulReplicaFlags**, but are otherwise ignored by the protocol:

* DRS\_NONGC\_RO\_REP: Replicate a read-only full replica. Not a writable or partial replica.
* DRS\_FULL\_SYNC\_IN\_PROGRESS: When the flag DRS\_FULL\_SYNC\_NOW is received in a call to IDL\_DRSReplicaSync, the flag DRS\_FULL\_SYNC\_IN\_PROGRESS is sent in the associated calls to IDL\_DRSGetNCChanges until the replication cycle completes. This flag is ignored by the server.
* DRS\_FULL\_SYNC\_PACKET: Replicate all [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) in the replication request, even those that would normally be filtered.
* DRS\_REF\_GCSPN: Requests that the server add an entry to repsTo for the client on the root object of the NC replica that is being replicated. When repsTo is set using this flag, the notifying client DC contacts the server DC using the [**service principal name**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) that begins with "GC" (section [2.2.3.2](#Section_41efc56e00074e88bafed7af61efd91f)).
* DRS\_NEVER\_SYNCED: There is no successfully completed replication from this source server.
* DRS\_SPECIAL\_SECRET\_PROCESSING: Do not replicate attribute values of attributes that contain [**secret data**](#gt_0c8d49b7-bdf7-4824-a91f-481cb10c5052).
* DRS\_PREEMPTED: The replication attempt is preempted by a higher priority replication request.
* DRS\_NEVER\_NOTIFY: Do not send update notifications.
* DRS\_SYNC\_PAS: Expand the [**partial attribute set**](#gt_2b3cc270-8a21-4402-bb8b-9bebac24bdaa) of the partial replica.

**schedule:** A [REPLTIMES](#Section_42d7d8e8794e427998028b5916e8b099) that corresponds to rtSchedule in REPS\_FROM. This contains the periodic replication schedule.

**uuidInvocId:** A GUID that contains the [**invocation ID**](#gt_e7869b9a-61fa-46e3-89dd-fb3f57d1ba7a) of the source DC.

**usnVec:** A [USN\_VECTOR](#Section_595d11b86ca74a61bd563e6a2b99b76b) that corresponds to the usnVec in REPS\_FROM. This holds 0 or the usnvecTo field from the response to the last IDL\_DRSGetNCChanges replication request sent to the DC identified by uuidDsa.

**uuidTransport:** A GUID that corresponds to uuidTransportObj in REPS\_FROM. This is the objectGUID of the interSiteTransport object that corresponds to the transport used for communication with the DC identified by uuidDsa.

**consecutiveFailures:** A [DWORD](#Section_60c3f5f194924d1083c89a155e162ef3) that corresponds to cConsecutiveFailures in REPS\_FROM. It is the number of consecutive failures during replication from the DC identified by uuidDsa.

**timeLastSuccess:** A DWORD that corresponds to timeLastSuccess in REPS\_FROM. It is the time of the last successful replication from the DC identified by uuidDsa.

**timeLastAttempt:** A DWORD that corresponds to timeLastAttempt in REPS\_FROM. It is the time of the last replication attempt with the DC identified by uuidDsa.

**resultLastAttempt:** The result of the last replication attempt with the DC identified by uuidDsa.

**pasData:** A [PAS\_DATA](#Section_14e0e4828f1d4fa589ded9fd7f98b10d) value that corresponds to **cbPasDataOffset** in REPS\_FROM. Contains the list of attributes (being added to the partial attribute set for the NC on this DC) that are being requested from the DC identified by uuidDsa as part of a PAS replication cycle.

When converting a RepsFrom to a REPS\_FROM, assign zeros to all unused fields of REPS\_FROM. If naDsa is an empty string, set cbOtherDra to 0 and cbOtherDraOffset to 0. If pasData.pas.cAttrs is 0, set cbPasDataOffset to 0.

## repsTo, RepsTo

The nonreplicated, multivalued [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) repsTo is an optional attribute on the root [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) of every [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210). It is stored as the structure [REPS\_TO](#Section_b422aa877d074527b070c5d719696c43).

The [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) RepsTo simplifies the specification of methods that read and write the attribute repsTo. Reading the attribute repsTo produces one or more RepsTo values using the conversions from REPS\_TO specified below. Writing a RepsTo value to the attribute repsTo stores a REPS\_TO using the reverse conversion.

The type RepsTo is a tuple with the following fields:

**naDsa:** A [NetworkAddress](#Section_3d0d3777935847ddb55534405f57f912) that corresponds to cbOtherDraOffset and cbOtherDra in REPS\_TO. This is the NetworkAddress of a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**uuidDsa:** A [GUID](#Section_5e740f50e6a048c9bca800072e85d963) that corresponds to uuidDsaObj in REPS\_TO. This is the [**DSA GUID**](#gt_736b1076-d1cb-40b9-9247-d66cca6819d1) of the target DC.

**options:** Bit flags chosen from [DRS\_OPTIONS](#Section_ac9c8a11cd464080acbf9faa86344030) that correspond to ulReplicaFlags in REPS\_TO. This set contains the DRS\_WRIT\_REP value if this [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) is writable.

The following additional values are preserved if they are present when reading ulReplicaFlags, but are otherwise ignored by the protocol:

* DRS\_INIT\_SYNC: The replica must be replicated from the DC identified by uuidDsa when the DC hosting this replica is started.
* DRS\_PER\_SYNC: Periodically replicate the NC replica from the DC identified by uuidDsa, as defined by the periodic [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) schedule.
* DRS\_MAIL\_REP: Replicate the NC replica from the DC identified by uuidDsa via SMTP (see [[MS-SRPL]](%5bMS-SRPL%5d.pdf#Section_ec69eea50d5e428ab5bc66732aaeb866)).
* DRS\_DISABLE\_AUTO\_SYNC: Disable notification-based replication of the NC replica from the DC identified by uuidDsa.
* DRS\_DISABLE\_PERIODIC\_SYNC: Disable periodic replication of the NC replica from the DC identified by uuidDsa.
* DRS\_USE\_COMPRESSION: Replication response messages sent along this communication path must be compressed.
* DRS\_TWOWAY\_SYNC: At the end of a [**replication cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16), replication must be triggered in the opposite direction.
* DRS\_NONGC\_RO\_REP: Replicate a read-only full replica. Not a writable or partial replica.
* DRS\_FULL\_SYNC\_IN\_PROGRESS: When the flag DRS\_FULL\_SYNC\_NOW is received in a call to IDL\_DRSReplicaSync, the flag DRS\_FULL\_SYNC\_IN\_PROGRESS is sent in the associated calls to IDL\_DRSGetNCChanges until the replication cycle completes. This flag is ignored by the server.
* DRS\_FULL\_SYNC\_PACKET: Replicate all [**updates**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) in the replication request, even those that would normally be filtered.
* DRS\_REF\_GCSPN: Requests that the server add an entry to repsTo for the client on the root object of the NC replica that is being replicated. When repsTo is set using this flag, the notifying client DC contacts the server DC using the [**service principal name**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) that begins with "GC" (section [2.2.3.2](#Section_41efc56e00074e88bafed7af61efd91f)).
* DRS\_NEVER\_SYNCED: There is no successfully completed replication from this source server.
* DRS\_SPECIAL\_SECRET\_PROCESSING: Do not replicate attribute values of attributes that contain [**secret data**](#gt_0c8d49b7-bdf7-4824-a91f-481cb10c5052).
* DRS\_PREEMPTED: The replication attempt is preempted by a higher priority replication request.
* DRS\_NEVER\_NOTIFY: Do not send update notifications.
* DRS\_SYNC\_PAS: Expand the [**partial attribute set**](#gt_2b3cc270-8a21-4402-bb8b-9bebac24bdaa) of the partial replica.

**resultLastAttempt:** A [DWORD](#Section_60c3f5f194924d1083c89a155e162ef3) that corresponds to ulResultLastAttempt in REPS\_TO. Contains the result of the last attempt to send a replication notification to the DC identified by uuidDsa. It has a value of 0 if the last notification was sent successfully and a [**Windows error code**](#gt_459db7bd-5066-44e3-89c1-f0e4806b7a1b) otherwise.

**consecutiveFailures:** A DWORD that corresponds to cConsecutiveFailures in REPS\_TO. Contains the number of unsuccessful consecutive attempts to send a replication notification to the DC identified by uuidDsa.

**timeLastAttempt:** A [DSTIME](#Section_a72a16b973e441caa5c1afc5fc54e175) that corresponds to timeLastAttempt in REPS\_TO. Contains the last time when an attempt was made to send a replication notification to the DC identified by uuidDsa, or 0 if no attempt has been made.

**timeLastSuccess:** A DSTIME that corresponds to timeLastSuccess in REPS\_TO. Contains the time when the last successful replication notification to the DC identified by uuidDsa was sent, or 0 if no replication notification has been successfully sent.

When converting a RepsTo to a REPS\_TO, assign zeros to all unused fields of REPS\_TO. If naDsa is an empty string, set cbOtherDra to 0 and cbOtherDraOffset to 0.

## Rid

Rid is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that consists of an integer that represents the [**relative identifier (RID)**](#gt_df3d0b61-56cd-4dac-9402-982f1fedc41c) component of a [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d), as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2.

## Right

Right is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that represents an access right (for example, RIGHT\_DS\_WRITE\_PROPERTY) or a [**control access right**](#gt_42f6c9e0-a2b3-4bc3-9b87-fdb902e5505e) (for example, DS-Replication-Manage-Topology) on an [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). The complete set of access right values is specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 5.1.3.2, and the complete set of control access right values is specified in [MS-ADTS] section 5.1.3.2.1.

**Note**  Since access rights and control access rights are non-overlapping sets, there is no ambiguity in having one type represent rights of both kinds.

## RIGHT Values

The valid access rights used in [**ACLs**](#gt_9f92aa05-dd0a-45f2-88d6-89f1fb654395) in [**security descriptors**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) are defined in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 5.1.3.2.

## RPCClientContexts

RPCClientContexts is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that is a sequence of tuples, one tuple per [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context for an incoming RPC session to the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). Each tuple contains the following fields:

* **BindingContext**: A [ULONGLONG](#Section_4e552c46a7dc4504a902f210e6e6dedd) that contains a unique identifier for the context.
* **RefCount**: An integer that is used to reference count the number of references to the context.
* **IsBound**: A Boolean value that is true if IDL\_DRSUnbind has not yet been called on the RPC context represented by this tuple, and false otherwise.
* **UUIDClient**: A [GUID](#Section_5e740f50e6a048c9bca800072e85d963) that contains the value that was passed in as the puuidClientDsa argument of IDL\_DRSBind while establishing the context.
* **TimeLastUsed**: A [FILETIME](#Section_70ee934bc9b944498aa36dfe9cef3eff) that contains the last time a session corresponding to the context was used in an RPC method call.
* **IPAddress**: A [DWORD](#Section_60c3f5f194924d1083c89a155e162ef3) that contains the IPv4 address of the client associated with the context.
* **PID**: An integer that contains the process ID passed in by the client as the pextClient argument of IDL\_DRSBind while establishing the context.

The global variable [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219) for a DC has an associated field dc.rpcClientContexts, which maintains the DC's RPCClientContexts state.

## RPCOutgoingContexts

RPCOutgoingContexts is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) that is a sequence of tuples, one tuple per [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context for an outgoing RPC session from the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). Each tuple contains the following fields:

* **ServerName**: A *unicodestring* (section [3.4.3](#Section_fbe9988847824858b5f25b521a44d836)) that contains the host name of the server.
* **IsBound**: A Boolean value that is true if IDL\_DRSUnbind has not yet been called on the RPC context represented by this tuple, and false otherwise.
* **HandleFromCache**: A Boolean value that is true if the context handle was retrieved from the cache, and false otherwise.
* **HandleInCache**: A Boolean value that is true if the context handle is still in the cache, and false otherwise.
* **ThreadId**: An integer that contains the thread ID of the thread that is using the context.
* **BindingTimeOut**: An integer. If the context is set to be canceled, then this field contains the time-out, in minutes.
* **CreateTime**: A [DSTIME](#Section_a72a16b973e441caa5c1afc5fc54e175) value that contains the time when the context was created.
* **CallType**: An integer that indicates the type of RPC call that the DC is waiting on. See DS\_REPL\_SERVER\_OUTGOING\_CALL for possible values.

The global variable [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219) for a DC has an associated field dc.rpcOutgoingContexts, which maintains the DC's RPCOutgoingContexts state.

## sAMAccountType Values

sAMAccountType values describe information about various account type [**objects**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca). See [[MS-SAMR]](%5bMS-SAMR%5d.pdf#Section_4df07fab1bbc452f8e927853a3c7e380) section 2.2.1.9 for descriptions of these values.

| Symbolic name | Value |
| --- | --- |
| SAM\_GROUP\_OBJECT | 0x10000000 |
| SAM\_NON\_SECURITY\_GROUP\_OBJECT | 0x10000001 |
| SAM\_ALIAS\_OBJECT | 0x20000000 |
| SAM\_NON\_SECURITY\_ALIAS\_OBJECT | 0x20000001 |
| SAM\_USER\_OBJECT | 0x30000000 |
| SAM\_MACHINE\_ACCOUNT | 0x30000001 |
| SAM\_TRUST\_ACCOUNT | 0x30000002 |

Only the values used by this protocol are contained in this table.

## SCHEMA\_PREFIX\_TABLE

The SCHEMA\_PREFIX\_TABLE structure defines the [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a table to map [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) values to and from [OID](#Section_339504853a964b668a28a3a33e80302b)s.

1. typedef struct {
2. [range(0,1048576)] DWORD PrefixCount;
3. [size\_is(PrefixCount)] PrefixTableEntry\* pPrefixEntry;
4. } SCHEMA\_PREFIX\_TABLE;

**PrefixCount:**  The number of items in the **pPrefixEntry** array.

**pPrefixEntry:**  An array of [PrefixTableEntry](#Section_d26d36cd10c44b27a84e98336abf357a) items in the table.

## SchemaInfo

1. procedure SchemaInfo(): sequence of BYTE

The SchemaInfo procedure returns the value of the schemaInfo attribute on the object [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).schemaNC or a default value if that attribute has no value.

1. if dc.schemaNC()!schemaInfo = null then
2. return (21 hexadecimal digits: "FF0000000000000000000000000000000000000000")
3. else
4. return dc.SchemaNC()!schemaInfo
5. endif

## SchemaNC

1. procedure SchemaNC(): DSName

The SchemaNC procedure returns the [DSName](#Section_a0d5477a522946b9890a54b924d487d1) of [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219).schemaNC.

## SchemaObj

1. procedure SchemaObj(att: ATTRTYP): DSName

Given the [ATTRTYP](#Section_9117312908e6497c8266b5ac0aa5f983) *att* of an attributeSchema or classSchema [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) on this [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), the SchemaObj procedure returns the [**dsname**](#gt_4d5e1f08-aa00-4dde-9411-7dd6e09ed85a) of the attributeSchema or the classSchema object.

1. return select one o from children SchemaNC()
2. where AttrtypFromSchemaObj(o) = att

## ServerExtensions

1. procedure ServerExtensions(hDrs: DRS\_HANDLE): DRS\_EXTENSIONS\_INT

The ServerExtensions procedure returns the server extensions presented in the IDL\_DRSBind call that created *hDrs*. Any fields not specified by the server in the *ppextServer*^ parameter of IDL\_DRSBind are set to 0.

## SID

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the Windows NT operating system **SID** structure, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2.

## SidFromStringSid

1. procedure SidFromStringSid(stringSID: unicodestring): SID

The SidFromStringSid procedure converts the string representation of a [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) specified in *stringSID* (for example, S-1-5-3) to the [SID](#Section_13560cc227ff43a09d6fd686bccc5f3c) type, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2. See [MS-DTYP] section 2.4.2.1 for the string representation of a SID.

## StampLessThanOrEqualUTD

1. procedure StampLessThanOrEqualUTD(
2. stamp: AttributeStamp,
3. utd: UPTODATE\_VECTOR\_V1\_EXT) : boolean

*Informative summary of behavior*: The StampLessThanOrEqualUTD procedure is used to determine if an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) has already replicated (or should have already replicated).

1. i: integer
3. for i := 0 to utd.cNumCursors - 1
4. if utd.rgCursors[i].uuidDsa = stamp.uuidOriginating) and
5. (utd.rgCursors[i].usn >= stamp.usnOriginating) then
6. return true
7. endif
8. endfor
9. return false

## StartsWith

1. procedure StartsWith(s: unicodestring, p: unicodestring): boolean

The StartsWith procedure returns true if the string *p* is a prefix of string *s* and returns false otherwise.

## StringSidFromSid

1. procedure StringSidFromSid(sid: SID): unicodestring

The StringSidFromSid procedure converts a binary [SID](#Section_13560cc227ff43a09d6fd686bccc5f3c) specified in *sid* to the string representation of a [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) (for example, S-1-5-3). See [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2.1 for the string representation of a SID.

## SubString

1. procedure SubString(
2. s: unicodestring, start: integer, length: integer): unicodestring

The SubString procedure returns the portion of *s* beginning at the zero-based index *start* and containing length characters. If *start* is less than zero or greater than *s*.*length*-1, returns null. If *length* + *start* is greater than *s*.*length*, then *length* is treated as if it equals *s*.*length* - *start*.

## Syntax

1. procedure Syntax(attr: ATTRTYP): AttributeSyntax

The Syntax procedure returns the syntax of the [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) *attr*.

## SYNTAX\_ADDRESS

The SYNTAX\_ADDRESS packet is the [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a sequence of bytes or [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) characters.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| dataLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| byteVal (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**dataLen (4 bytes):** The size of the entire structure (including this field), in bytes.

**byteVal (variable):** The byte or character data.

The following structure definition shows an alternative representation of this data type.

1. typedef struct {
2. DWORD dataLen;
3. union {
4. BYTE byteVal[];
5. wchar\_t uVal[];
6. };
7. } SYNTAX\_ADDRESS;

## SYNTAX\_DISTNAME\_BINARY

The SYNTAX\_DISTNAME\_BINARY packet is the [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a combination of a [DSNAME](#Section_385d478f3eb64d2cac58f25c4debdd86) and a binary or character data buffer.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| structLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SidLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Guid (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Sid (28 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| NameLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| StringName (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Padding (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| dataLen | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| byteVal (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**structLen (4 bytes):** The length of the structure, in bytes, up to and including the field StringName.

**SidLen (4 bytes):** The number of bytes in the [**Sid**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) field used to represent the [**object's**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) objectSid [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) value. Zero indicates that the SYNTAX\_DISTNAME\_BINARY does not identify the objectSid value of the [**directory object**](#gt_407dbc2c-3140-4e31-9085-0087e2d3bab2).

**Guid (16 bytes):** The value of the object's objectGUID attribute specified as a [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) structure, which is defined in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.3.4. If the values for all fields in the GUID structure are zero, this indicates that the SYNTAX\_DISTNAME\_BINARY does not identify the objectGUID value of the directory object.

**Sid (28 bytes):** The value of the object's objectSid attribute, its security identifier specified as a SID structure, which is defined in [MS-DTYP] section 2.4.2. The size of this field is exactly 28 bytes, regardless of the value of SidLen, which specifies how many bytes in this field are used.

**NameLen (4 bytes):** The number of characters in the StringName field, not including the terminating null character, used to represent the object's distinguishedName attribute value. Zero indicates that the SYNTAX\_DISTNAME\_BINARY does not identify the distinguishedName value of the directory object.

**StringName (variable):** The null-terminated [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) value of the object's distinguishedName attribute, as specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.1.4. This field always contains at least one character: the terminating null character. Each Unicode value is encoded as 2 bytes. The byte ordering is little-endian.

**Padding (variable):** The padding (bytes with value zero) to align the field dataLen at a double word boundary.

**dataLen (4 bytes):** The length of the remaining structure, including this field, in bytes.

**byteVal (variable):** An array of bytes.

**Note**  All fields have little-endian byte ordering.

The following structure definition shows an alternative representation of this data type.

1. typedef struct {
2. DSNAME Name;
3. SYNTAX\_ADDRESS Data;
4. } SYNTAX\_DISTNAME\_BINARY;

## systemFlags Values

The valid system flags used on [**directory objects**](#gt_407dbc2c-3140-4e31-9085-0087e2d3bab2) are defined in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 2.2.10.

## UCHAR

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d), as defined in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.45. A UCHAR is an 8-bit, unsigned quantity.

## ULARGE\_INTEGER

ULARGE\_INTEGER is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a 64-bit, unsigned integer.

1. typedef struct {
2. ULONGLONG QuadPart;
3. } ULARGE\_INTEGER;

## ULONG

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a 32-bit, unsigned integer, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.51.

## ULONGLONG

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a 64-bit, unsigned integer, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.55.

## UndeleteObject

1. procedure UndeleteObject(
2. obj: DSNAME,
3. attributesAndStamps: set of AttributeAndStamp)

For each attStamp in *attributesAndStamps*, the UndeleteObject procedure performs an [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) to *obj* such that the value(s) of attStamp.[**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) do not change, but AttrStamp(obj, attStamp.attribute).dwVersion > attStamp.stamp.dwVersion. The effect of this [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493) to *obj* is such that this [DC's](#Section_c7f1df49ea514e1aa8af063ac3f5e219) values for these attributes replicate out to other DCs and overwrite the updates with [**stamps**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) in *attributesAndStamps*.

## UnbindFromDSA()

1. procedure UnbindFromDSA(hDRS: DRS\_HANDLE)

The UnbindFromDSA procedure closes the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) connection that was established by the [BindToDSA](#Section_032c290778c745d2acf8ecf0e98bad3b) procedure.

## UpdateRefs

1. procedure UpdateRefs(DRS\_MSG\_UPDREFS\_V1 msgIn): ULONG

The UpdateRefs method implements the core functionality of [IDL\_DRSUpdateRefs](#Section_a273bbcfaeca46088ad4127d3e597cd4), that is, adds or deletes a value from the repsTo of a specified [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210).

1. err: DWORD
2. nc: DSName
3. rt: RepsTo
4. nc := msgIn.pNC^
5. /\* If ulOptions contains DRS\_ASYNC\_OP, the server processes the request asynchronously.
6. if DRS\_ASYNC\_OP in msgIn.ulOptions then
7. Asynchronous Processing: Initiate a logical thread of control
8. to process the remainder of this request asynchronously
9. return ERROR\_SUCCESS
10. endif
11. /\* If DRS\_DEL\_REF is specified, the return value is that associated with the DRS\_DEL\_REF
12. if DRS\_DEL\_REF in msgIn.ulOptions then
13. rt := select one v from nc!repsTo where
14. (v.naDsa = msgIn.pszDsaDest or
15. v.uuidDsa = msgIn.uuidDsaObjDest)
16. if rt = null then
17. err := ERROR\_DS\_DRA\_REF\_NOT\_FOUND
18. else
19. nc!repsTo := nc!repsTo - {rt}
20. err := ERROR\_SUCCESS
21. endif
22. endif
23. /\* If DRS\_DEL\_REF and DRS\_ADD\_REF are both specified, the return
24. \* value is that associated with the DRS\_ADD\_REF. \*/
25. if DRS\_ADD\_REF in msgIn.ulOptions then
26. rt := select one v from nc!repsTo where
27. (v.naDsa = msgIn.pszDsaDest or
28. v.uuidDsa = msgIn.uuidDsaObjDest)
29. if rt = null then
30. rt.naDsa := msgIn.pszDsaDest
31. rt.uuidDsa := msgIn.uuidDsaObjDest
32. rt.options := msgIn.ulOptions ∩ {DRS\_WRIT\_REP}
33. rt.timeLastAttempt := 0
34. rt.timeLastSuccess := current time
35. rt.consecutiveFailures := 0
36. rt.resultLastAttempt := 0
37. nc!repsTo := nc!repsTo + {rt}
38. err := ERROR\_SUCCESS
39. else
40. err := ERROR\_DS\_DRA\_REF\_ALREADY\_EXISTS
41. endif
42. endif
43. return err

## UPTODATE\_CURSOR\_V1

The UPTODATE\_CURSOR\_V1 structure is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state relative to a given [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

1. typedef struct {
2. UUID uuidDsa;
3. USN usnHighPropUpdate;
4. } UPTODATE\_CURSOR\_V1;

**uuidDsa:**  The invocationId of the DC performing the [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493).

**usnHighPropUpdate:**  The [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) of the update on the updating DC.

A cursor c with c.uuidDsa = x and c.usnHighPropUpdate = y indicates a replication state that includes all changes originated by DC x at USN less than or equal to y.

## UPTODATE\_CURSOR\_V2

The UPTODATE\_CURSOR\_V2 structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state relative to a given [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

1. typedef struct {
2. UUID uuidDsa;
3. USN usnHighPropUpdate;
4. DSTIME timeLastSyncSuccess;
5. } UPTODATE\_CURSOR\_V2;

**uuidDsa:**  The invocationId of the DC performing the [**update**](#gt_b242435b-73cc-4c4e-95f0-b2a2ff680493).

**usnHighPropUpdate:**  The [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) of the update on the updating DC.

**timeLastSyncSuccess:**  The time at which the last successful replication occurred from the DC identified by **uuidDsa**; for [**replication latency**](#gt_2352e9b3-ae08-4b5f-8858-bbca4ff4dd97) reporting only.

A cursor c with c.uuidDsa = x and c.usnHighPropUpdate = y indicates a replication state that includes all changes originated by DC x at USN less than or equal to y.

## UPTODATE\_VECTOR\_V1\_EXT

The UPTODATE\_VECTOR\_V1\_EXT structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state relative to a set of [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

1. typedef struct {
2. DWORD dwVersion;
3. DWORD dwReserved1;
4. [range(0,1048576)] DWORD cNumCursors;
5. DWORD dwReserved2;
6. [size\_is(cNumCursors)] UPTODATE\_CURSOR\_V1 rgCursors[];
7. } UPTODATE\_VECTOR\_V1\_EXT;

**dwVersion:**  The version of this structure; MUST be 1.

**dwReserved1:**  Unused. MUST be 0 and ignored.

**cNumCursors:**  The number of items in the **rgCursors** array.

**dwReserved2:**  Unused. MUST be 0 and ignored.

**rgCursors:**  An array of [UPTODATE\_CURSOR\_V1](#Section_cf88f341fb494cd5b7e26920cbd91f1b). The items in this field MUST be sorted in increasing order of the **uuidDsa** field.

## UPTODATE\_VECTOR\_V2\_EXT

The UPTODATE\_VECTOR\_V2\_EXT structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) state relative to a set of [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

1. typedef struct {
2. DWORD dwVersion;
3. DWORD dwReserved1;
4. [range(0,1048576)] DWORD cNumCursors;
5. DWORD dwReserved2;
6. [size\_is(cNumCursors)] UPTODATE\_CURSOR\_V2 rgCursors[];
7. } UPTODATE\_VECTOR\_V2\_EXT;

**dwVersion:**  The version of this structure; MUST be 2.

**dwReserved1:**  Unused. MUST be 0 and ignored.

**cNumCursors:**  The number of items in the **rgCursors** array.

**dwReserved2:**  Unused. MUST be 0 and ignored.

**rgCursors:**  An array of [UPTODATE\_CURSOR\_V2](#Section_d3e30021b6ac413eb08ab69b9b0c6592). The items in this field MUST be sorted in increasing order of the **uuidDsa** field.

## userAccountControl Bits

The userAccountControl bits are bit flags that describe various qualities of a security account. The bit flags are presented below in little-endian byte order.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| X | X | X | L O | X | X | A D | X | X | X | S T | W T | I D | X | N A | D A | X | X | X | X | X | X | X | X | X | X | X | X | X | P S | X | X |

**X**: Unused. MUST be zero and ignored.

**AD (ADS\_UF\_ACCOUNTDISABLE, 0x00000002)**: The account is disabled.

**LO (ADS\_UF\_LOCKOUT, 0x00000010)**: The account is temporarily locked out.

**DA (ADS\_UF\_TEMP\_DUPLICATE\_ACCOUNT, 0x00000100)**: This is an account for a user whose primary account is in another [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

**NA (ADS\_UF\_NORMAL\_ACCOUNT, 0x00000200)**: The default account type that represents a typical user.

**ID (ADS\_UF\_INTERDOMAIN\_TRUST\_ACCOUNT, 0x00000800)**: The account for a domain-to-domain trust.

**WT (ADS\_UF\_WORKSTATION\_ACCOUNT, 0x00001000)**: The computer account for a computer that is a member of this domain.

**ST (ADS\_UF\_SERVER\_TRUST\_ACCOUNT, 0x00002000)**: The computer account for a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**PS (ADS\_UF\_PARTIAL\_SECRETS\_ACCOUNT, 0x04000000)**: The computer account for an [**RODC**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870).

## UserNameFromNT4AccountName

1. procedure UserNameFromNT4AccountName(
2. nt4AccountName: unicodestring): unicodestring

If *nt4AccountName* is a name in Windows NT 4.0 account name format, that is, two components separated by a backslash (for example, "DOMAIN\username"), the UserNameFromNT4AccountName procedure returns the second component (the user name, or "username" in this example). If the *nt4AccountName* is not in this format, null is returned.

## USHORT

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for a 16-bit, unsigned integer, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.58.

## USN

[**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c) is a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the variable *usn* specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.1.9 and present in the [dc](#Section_c7f1df49ea514e1aa8af063ac3f5e219) global variable.

This type is declared as follows:

1. typedef LONGLONG USN;

## USN\_VECTOR

The USN\_VECTOR structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the cookie (section [1.3.2](#Section_67c5a415a6c740988cf36ef8d173cfe8)) used to pass state between calls to IDL\_DRSGetNCChanges.

1. typedef struct {
2. USN usnHighObjUpdate;
3. USN usnReserved;
4. USN usnHighPropUpdate;
5. } USN\_VECTOR;

**usnHighObjUpdate:**  A [**USN**](#gt_01936446-8739-4b98-b83f-fb5e2a53ce4c).

**usnReserved:**  A USN.

**usnHighPropUpdate:**  A USN.

The USN\_VECTOR type, as shown, is used in the DRS [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824). However, only the size of USN\_VECTOR (24 bytes) and the representation of its null value (24 zero bytes) are standardized for interoperability.

## UUID

[**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) is a type that is equivalent to the [GUID](#Section_5e740f50e6a048c9bca800072e85d963) type.

## ValidateDRSDemotionInput

1. procedure ValidateDRSDemotionInput(hDrs: DRS\_HANDLE, opnum: integer)

*Informative summary of behavior*: The ValidateDRSDemotionInput procedure performs certain checks based on the input and throws an exception, if needed.

The server MUST raise an ERROR\_INVALID\_PARAMETER exception when *opnum* = 25 and IsAdlds() == false.

The server SHOULD raise an ERROR\_INVALID\_PARAMETER exception when *opnum* = 26 and IsAdlds() == false.[<51>](#Appendix_A_51" \o "Product behavior note 51)

The server MUST raise an ERROR\_INVALID\_PARAMETER exception when *opnum* = 27 and IsAdlds() == false.

## ValidateDRSInput

1. procedure ValidateDRSInput(hDrs: DRS\_HANDLE, opnum: integer)

*Informative summary of behavior*: The ValidateDRSInput procedure performs certain checks based on the input and throws an exception, if needed.

1. if opnum = 0 then
2. return
3. endif
4. if (hDrs = null) then
5. raise ERROR\_INVALID\_HANDLE exception
6. endif
7. if (ClientUUID(hDrs) ≠ NTDSAPI\_CLIENT\_GUID) and
8. (IsServerExtensionsChanged(ServerExtensions(hDrs)) and
9. opnum ≠ 1
10. then
11. raise ERROR\_DS\_DRS\_EXTENSIONS\_CHANGED exception
12. endif
13. if (ClientUUID(hDrs) ≠ NTDSAPI\_CLIENT\_GUID) and
14. (ClientExtensions(hDrs).dwReplEpoch ≠ DSAObj()!msDS-ReplicationEpoch) and
15. opnum ≠ 1
16. then
17. raise ERROR\_DS\_DIFFERENT\_REPL\_EPOCHS exception
18. endif
19. if IsAdlds() and
20. (opnum = 9 or /\*IDL\_DRSGetMemberships\*/
21. opnum = 10 or /\*IDL\_DRSInterDomainMove\*/
22. opnum = 11 or /\*IDL\_DRSGetNT4ChangeLog\*/
23. opnum = 13 or /\*IDL\_DRSWriteSPN\*/
24. opnum = 15 or /\*IDL\_DRSRemoveDsDomain\*/
25. opnum = 16 or /\*IDL\_DRSDomainControllerInfo\*/
26. opnum = 20 or /\*IDL\_DRSAddSidHistory\*/
27. opnum = 21 or /\*IDL\_DRSGetMemberships2\*/
28. opnum = 24 /\*IDL\_DRSQuerySitesByCost\*/)
29. then
30. raise ERROR\_INVALID\_PARAMETER exception
31. endif
32. if AmIRODC() and
33. (opnum = 3 or /\*IDL\_DRSGetNCChanges\*/
34. opnum = 10 or /\*IDL\_DRSInterDomainMove\*/
35. opnum = 11 or /\*IDL\_DRSGetNT4ChangeLog\*/
36. opnum = 14 or /\*IDL\_DRSRemoveDsServer\*/
37. opnum = 15 or /\*IDL\_DRSRemoveDsDomain\*/
38. opnum = 17 or /\*IDL\_DRSAddEntry\*/
39. opnum = 20 /\*IDL\_DRSAddSidHistory\*/)
40. then
41. raise ERROR\_INVALID\_PARAMETER exception
42. endif
43. ValidateDRSDemotionInput(hDrs, opnum)

## Value

Value is an [**abstract type**](#gt_670b9ecd-79c3-4edc-8aca-aae65c83879b) for [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f) values used for abstract value representation (see section [5.16.2](#Section_284c8a5a6ede4d3488babda0b8bb59e0)).

## VALUE\_META\_DATA\_EXT\_V1

The VALUE\_META\_DATA\_EXT\_V1 structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) of a [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238).

1. typedef struct {
2. DSTIME timeCreated;
3. PROPERTY\_META\_DATA\_EXT MetaData;
4. } VALUE\_META\_DATA\_EXT\_V1;

**timeCreated:**  The date and time at which the first [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) was made.

**MetaData:**  The remainder of the stamp; has the same PROPERTY\_META\_DATA\_EXT type as used for the stamp of an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).

## VALUE\_META\_DATA\_EXT\_V3

The VALUE\_META\_DATA\_EXT\_V3 structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) for the [**stamp**](#gt_ff635a35-a17d-477b-a30d-9723b415bf00) of a [**link value**](#gt_659e8352-a6db-4752-8c05-4b21c602f238). This structure is a superset of the [VALUE\_META\_DATA\_EXT\_V1](#Section_7530cf2ea2ad4716a5708383f8b1846f) structure.

1. typedef struct {
2. DSTIME timeCreated;
3. PROPERTY\_META\_DATA\_EXT MetaData;
4. DWORD unused1;
5. DWORD unused2;
6. DWORD unused3;
7. DSTIME timeExpired;
8. } VALUE\_META\_DATA\_EXT\_V3;

**timeCreated**: The date and time at which the first [**originating update**](#gt_119f7bf0-9100-4f4a-b593-ab4e6ccfea20) was made.

**MetaData**: The remainder of the stamp; has the same PROPERTY\_META\_DATA\_EXT type as used for the stamp of an [**attribute**](#gt_108a1419-49a9-4d19-b6ca-7206aa726b3f).

**unused1**: Unused. MUST be 0 and ignored

**unused2**: Unused. MUST be 0 and ignored

**unused3**: Unused. MUST be 0 and ignored

**timeExpired**: The date and time at which the link value must be removed from the state of the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

## VALUE\_META\_DATA\_EXT\_NATIVE

The VALUE\_META\_DATA\_EXT\_NATIVE structure is an alias for the [**VALUE\_META\_DATA\_EXT\_V3**](#Section_eab72899a828427d83849a51ffdb77e1) data structure.

## ValueFromATTRVAL

1. procedure ValueFromATTRVAL(
2. a: ATTRVAL, s: Syntax, t: PrefixTable) : Value

The ValueFromATTRVAL procedure converts a value of syntax *s* expressed as a concrete [ATTRVAL](#Section_cc002cbfefe042f89295a5a6577263d4) *a* into the abstract [Value](#Section_c1b732d37bf94ba181ee07157f07294c) encoding, using the [**prefix table**](#gt_028437b6-7749-4428-b874-22e9559c1abe) represented by *t*.

See section [5.16.3](#Section_0d7070d2f71647109f92812dc4cd8a53) for the specification of this procedure.

## VAR\_SIZE\_BUFFER\_WITH\_VERSION

The VAR\_SIZE\_BUFFER\_WITH\_VERSION structure defines a [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) that is used to pass byte buffers to certain messages. The exact byte pattern is dependent on the structure in which this structure is being used.

1. typedef struct \_VAR\_SIZE\_BUFFER\_WITH\_VERSION
2. {
3. ULONG ulVersion;
4. ULONG cbByteBuffer;
5. ULONGLONG ullPadding;
6. [size\_is(cbByteBuffer)] BYTE rgbBuffer[];
7. } VAR\_SIZE\_BUFFER\_WITH\_VERSION;

**ulVersion**: The version of the buffer that is being sent. Handling of this field is performed by the specific message that is using this structure.

**cbByteBuffer**: The size, in bytes, of the data in the **rgbBuffer** field.

**ullPadding**: Used to align the array of bytes in the **rgbBuffer** field to an 8-byte boundary.

**rgbBuffer**: An array of bytes. The content of the array depends on the specific message that is using this structure. Starts on an 8-byte boundary.

## WCHAR

A [**concrete type**](#gt_cd539538-9f7e-4881-b5af-2301b420244d), as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.60. A WCHAR is a 16-bit, unsigned integer in little-endian byte order that is used to store a double-byte [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) character. A WCHAR \* is a pointer to a null-terminated Unicode string.

# Security

## Security Considerations for Implementers

General security considerations for this protocol are specified in section [2.2](#Section_0a156712918047bab0802e285f127a7f). Security considerations for an individual method are specified in the subsection of section [4](#Section_9554afa5e7554742a34b899fc4e2fd20) that describes the behavior of that method.

## Index of Security Parameters

| Security parameter | Section |
| --- | --- |
| [**SPNs**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4) for [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd)-to-DC [**authentication**](#gt_8e961bf0-95ba-4f58-9034-b67ccb27f317) | Section [2.2.3.2](#Section_41efc56e00074e88bafed7af61efd91f) |
| SPNs for client-to-DC authentication | Section [2.2.4.2](#Section_894d09997d794e81a4077bcf6522b0a7) |

# Appendix A: Full IDL

For ease of implementation, the full IDL is provided below, where "ms-dtyp.idl" refers to the IDL found in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) Appendix A.

1. import "ms-dtyp.idl";
2. [
3. uuid (e3514235-4b06-11d1-ab04-00c04fc2dcd2), version(4.0),
4. pointer\_default (unique)
5. ]
6. interface drsuapi
7. {
8. typedef LONGLONG DSTIME;
9. typedef [context\_handle] void \* DRS\_HANDLE;
10. typedef struct {
11. char Data[28];
12. } NT4SID;
13. typedef struct {
14. unsigned long structLen;
15. unsigned long SidLen;
16. GUID Guid;
17. NT4SID Sid;
18. unsigned long NameLen;
19. [range(0, 10485761)] [size\_is(NameLen + 1)] WCHAR StringName[];
20. } DSNAME;
21. typedef LONGLONG USN;
22. typedef struct {
23. USN usnHighObjUpdate;
24. USN usnReserved;
25. USN usnHighPropUpdate;
26. } USN\_VECTOR;
27. typedef struct {
28. UUID uuidDsa;
29. USN usnHighPropUpdate;
30. } UPTODATE\_CURSOR\_V1;
31. typedef struct {
32. DWORD dwVersion;
33. DWORD dwReserved1;
34. [range(0,1048576)] DWORD cNumCursors;
35. DWORD dwReserved2;
36. [size\_is(cNumCursors)] UPTODATE\_CURSOR\_V1 rgCursors[];
37. } UPTODATE\_VECTOR\_V1\_EXT;
38. typedef struct {
39. [range(0,10000)] unsigned int length;
40. [size\_is(length)] BYTE \*elements;
41. } OID\_t;
42. typedef struct {
43. unsigned long ndx;
44. OID\_t prefix;
45. } PrefixTableEntry;
46. typedef struct {
47. [range(0,1048576)] DWORD PrefixCount;
48. [size\_is(PrefixCount)] PrefixTableEntry \*pPrefixEntry;
49. } SCHEMA\_PREFIX\_TABLE;
50. typedef ULONG ATTRTYP;
51. typedef struct {
52. DWORD dwVersion;
53. DWORD dwReserved1;
54. [range(1,1048576)] DWORD cAttrs;
55. [size\_is(cAttrs)] ATTRTYP rgPartialAttr[];
56. } PARTIAL\_ATTR\_VECTOR\_V1\_EXT;
57. typedef struct {
58. [range(1,256)] unsigned long mtx\_namelen;
59. [size\_is(mtx\_namelen)] char mtx\_name[];
60. } MTX\_ADDR;
61. typedef struct {
62. [range(0,26214400)] ULONG valLen;
63. [size\_is(valLen)] UCHAR \*pVal;
64. } ATTRVAL;
65. typedef struct {
66. [range(0, 10485760)] ULONG valCount;
67. [size\_is(valCount)] ATTRVAL \*pAVal;
68. } ATTRVALBLOCK;
69. typedef struct {
70. ATTRTYP attrTyp;
71. ATTRVALBLOCK AttrVal;
72. } ATTR;
73. typedef struct {
74. [range(0, 1048576)] ULONG attrCount;
75. [size\_is(attrCount)] ATTR \*pAttr;
76. } ATTRBLOCK;
77. typedef struct {
78. DSNAME \*pName;
79. unsigned long ulFlags;
80. ATTRBLOCK AttrBlock;
81. } ENTINF;
82. typedef struct {
83. DWORD dwVersion;
84. DSTIME timeChanged;
85. UUID uuidDsaOriginating;
86. USN usnOriginating;
87. } PROPERTY\_META\_DATA\_EXT;
88. typedef struct {
89. [range(0,1048576)] DWORD cNumProps;
90. [size\_is(cNumProps)] PROPERTY\_META\_DATA\_EXT rgMetaData[];
91. } PROPERTY\_META\_DATA\_EXT\_VECTOR;
92. typedef struct REPLENTINFLIST {
93. struct REPLENTINFLIST \* pNextEntInf;
94. ENTINF Entinf;
95. BOOL fIsNCPrefix;
96. UUID\* pParentGuid;
97. PROPERTY\_META\_DATA\_EXT\_VECTOR\* pMetaDataExt;
98. } REPLENTINFLIST;
99. typedef struct {
100. UUID uuidDsa;
101. USN usnHighPropUpdate;
102. DSTIME timeLastSyncSuccess;
103. } UPTODATE\_CURSOR\_V2;
104. typedef struct {
105. DWORD dwVersion;
106. DWORD dwReserved1;
107. [range(0,1048576)] DWORD cNumCursors;
108. DWORD dwReserved2;
109. [size\_is(cNumCursors)] UPTODATE\_CURSOR\_V2 rgCursors[];
110. } UPTODATE\_VECTOR\_V2\_EXT;
111. typedef struct {
112. DSTIME timeCreated;
113. PROPERTY\_META\_DATA\_EXT MetaData;
114. } VALUE\_META\_DATA\_EXT\_V1;
115. typedef struct {
116. DSTIME timeCreated;
117. PROPERTY\_META\_DATA\_EXT MetaData;
118. DWORD unused1;
119. DWORD unused2;
120. DWORD unused3;
121. DSTIME timeExpired;
122. } VALUE\_META\_DATA\_EXT\_V3;
123. typedef struct {
124. DSNAME \*pObject;
125. ATTRTYP attrTyp;
126. ATTRVAL Aval;
127. BOOL fIsPresent;
128. VALUE\_META\_DATA\_EXT\_V1 MetaData;
129. } REPLVALINF\_V1;
130. typedef struct {
131. DSNAME \*pObject;
132. ATTRTYP attrTyp;
133. ATTRVAL Aval;
134. BOOL fIsPresent;
135. VALUE\_META\_DATA\_EXT\_V3 MetaData;
136. } REPLVALINF\_V3;
137. typedef struct {
138. UCHAR rgTimes[84];
139. } REPLTIMES;
140. typedef struct {
141. DWORD status;
142. [string,unique] WCHAR \*pDomain;
143. [string,unique] WCHAR \*pName;
144. } DS\_NAME\_RESULT\_ITEMW, \*PDS\_NAME\_RESULT\_ITEMW;
145. typedef struct {
146. DWORD cItems;
147. [size\_is(cItems)] PDS\_NAME\_RESULT\_ITEMW rItems;
148. } DS\_NAME\_RESULTW, \*PDS\_NAME\_RESULTW;
149. typedef struct {
150. [string,unique] WCHAR \*NetbiosName;
151. [string,unique] WCHAR \*DnsHostName;
152. [string,unique] WCHAR \*SiteName;
153. [string,unique] WCHAR \*ComputerObjectName;
154. [string,unique] WCHAR \*ServerObjectName;
155. BOOL fIsPdc;
156. BOOL fDsEnabled;
157. } DS\_DOMAIN\_CONTROLLER\_INFO\_1W;
158. typedef struct {
159. [string,unique] WCHAR \*NetbiosName;
160. [string,unique] WCHAR \*DnsHostName;
161. [string,unique] WCHAR \*SiteName;
162. [string,unique] WCHAR \*SiteObjectName;
163. [string,unique] WCHAR \*ComputerObjectName;
164. [string,unique] WCHAR \*ServerObjectName;
165. [string,unique] WCHAR \*NtdsDsaObjectName;
166. BOOL fIsPdc;
167. BOOL fDsEnabled;
168. BOOL fIsGc;
169. GUID SiteObjectGuid;
170. GUID ComputerObjectGuid;
171. GUID ServerObjectGuid;
172. GUID NtdsDsaObjectGuid;
173. } DS\_DOMAIN\_CONTROLLER\_INFO\_2W;
174. typedef struct {
175. [string, unique] WCHAR\* NetbiosName;
176. [string, unique] WCHAR\* DnsHostName;
177. [string, unique] WCHAR\* SiteName;
178. [string, unique] WCHAR\* SiteObjectName;
179. [string, unique] WCHAR\* ComputerObjectName;
180. [string, unique] WCHAR\* ServerObjectName;
181. [string, unique] WCHAR\* NtdsDsaObjectName;
182. BOOL fIsPdc;
183. BOOL fDsEnabled;
184. BOOL fIsGc;
185. BOOL fIsRodc;
186. GUID SiteObjectGuid;
187. GUID ComputerObjectGuid;
188. GUID ServerObjectGuid;
189. GUID NtdsDsaObjectGuid;
190. } DS\_DOMAIN\_CONTROLLER\_INFO\_3W;
191. typedef struct {
192. DWORD IPAddress;
193. DWORD NotificationCount;
194. DWORD secTimeConnected;
195. DWORD Flags;
196. DWORD TotalRequests;
197. DWORD Reserved1;
198. [string,unique] WCHAR \*UserName;
199. } DS\_DOMAIN\_CONTROLLER\_INFO\_FFFFFFFFW;
200. typedef struct ENTINFLIST {
201. struct ENTINFLIST \*pNextEntInf;
202. ENTINF Entinf;
203. } ENTINFLIST;
204. typedef struct {
205. DWORD dsid;
206. DWORD extendedErr;
207. DWORD extendedData;
208. USHORT problem;
209. ATTRTYP type;
210. BOOL valReturned;
211. ATTRVAL Val;
212. } INTFORMPROB\_DRS\_WIRE\_V1;
213. typedef struct \_PROBLEMLIST\_DRS\_WIRE\_V1 {
214. struct \_PROBLEMLIST\_DRS\_WIRE\_V1 \*pNextProblem;
215. INTFORMPROB\_DRS\_WIRE\_V1 intprob;
216. } PROBLEMLIST\_DRS\_WIRE\_V1;
217. typedef struct {
218. DSNAME \*pObject;
219. ULONG count;
220. PROBLEMLIST\_DRS\_WIRE\_V1 FirstProblem;
221. } ATRERR\_DRS\_WIRE\_V1;
222. typedef struct {
223. DWORD dsid;
224. DWORD extendedErr;
225. DWORD extendedData;
226. USHORT problem;
227. DSNAME \*pMatched;
228. } NAMERR\_DRS\_WIRE\_V1;
229. typedef struct {
230. UCHAR nameRes;
231. UCHAR unusedPad;
232. USHORT nextRDN;
233. } NAMERESOP\_DRS\_WIRE\_V1;
234. typedef struct \_DSA\_ADDRESS\_LIST\_DRS\_WIRE\_V1 {
235. struct \_DSA\_ADDRESS\_LIST\_DRS\_WIRE\_V1 \*pNextAddress;
236. RPC\_UNICODE\_STRING \*pAddress;
237. } DSA\_ADDRESS\_LIST\_DRS\_WIRE\_V1;
238. typedef struct CONTREF\_DRS\_WIRE\_V1 {
239. DSNAME \*pTarget;
240. NAMERESOP\_DRS\_WIRE\_V1 OpState;
241. USHORT aliasRDN;
242. USHORT RDNsInternal;
243. USHORT refType;
244. USHORT count;
245. DSA\_ADDRESS\_LIST\_DRS\_WIRE\_V1 \*pDAL;
246. struct CONTREF\_DRS\_WIRE\_V1 \*pNextContRef;
247. BOOL bNewChoice;
248. UCHAR choice;
249. } CONTREF\_DRS\_WIRE\_V1;
250. typedef struct {
251. DWORD dsid;
252. DWORD extendedErr;
253. DWORD extendedData;
254. CONTREF\_DRS\_WIRE\_V1 Refer;
255. } REFERR\_DRS\_WIRE\_V1;
256. typedef struct {
257. DWORD dsid;
258. DWORD extendedErr;
259. DWORD extendedData;
260. USHORT problem;
261. } SECERR\_DRS\_WIRE\_V1;
262. typedef struct {
263. DWORD dsid;
264. DWORD extendedErr;
265. DWORD extendedData;
266. USHORT problem;
267. } SVCERR\_DRS\_WIRE\_V1;
268. typedef struct {
269. DWORD dsid;
270. DWORD extendedErr;
271. DWORD extendedData;
272. USHORT problem;
273. } UPDERR\_DRS\_WIRE\_V1;
274. typedef struct {
275. DWORD dsid;
276. DWORD extendedErr;
277. DWORD extendedData;
278. USHORT problem;
279. } SYSERR\_DRS\_WIRE\_V1;
280. typedef [switch\_type(DWORD)] union {
281. [case(1)] ATRERR\_DRS\_WIRE\_V1 AtrErr;
282. [case(2)] NAMERR\_DRS\_WIRE\_V1 NamErr;
283. [case(3)] REFERR\_DRS\_WIRE\_V1 RefErr;
284. [case(4)] SECERR\_DRS\_WIRE\_V1 SecErr;
285. [case(5)] SVCERR\_DRS\_WIRE\_V1 SvcErr;
286. [case(6)] UPDERR\_DRS\_WIRE\_V1 UpdErr;
287. [case(7)] SYSERR\_DRS\_WIRE\_V1 SysErr;
288. } DIRERR\_DRS\_WIRE\_V1;
289. typedef struct {
290. [string] LPWSTR pszNamingContext;
291. [string] LPWSTR pszSourceDsaDN;
292. [string] LPWSTR pszSourceDsaAddress;
293. [string] LPWSTR pszAsyncIntersiteTransportDN;
294. DWORD dwReplicaFlags;
295. DWORD dwReserved;
296. UUID uuidNamingContextObjGuid;
297. UUID uuidSourceDsaObjGuid;
298. UUID uuidSourceDsaInvocationID;
299. UUID uuidAsyncIntersiteTransportObjGuid;
300. USN usnLastObjChangeSynced;
301. USN usnAttributeFilter;
302. FILETIME ftimeLastSyncSuccess;
303. FILETIME ftimeLastSyncAttempt;
304. DWORD dwLastSyncResult;
305. DWORD cNumConsecutiveSyncFailures;
306. } DS\_REPL\_NEIGHBORW;
307. typedef struct {
308. DWORD cNumNeighbors;
309. DWORD dwReserved;
310. [size\_is(cNumNeighbors)] DS\_REPL\_NEIGHBORW rgNeighbor[];
311. } DS\_REPL\_NEIGHBORSW;
312. typedef struct {
313. UUID uuidSourceDsaInvocationID;
314. USN usnAttributeFilter;
315. } DS\_REPL\_CURSOR;
316. typedef struct {
317. DWORD cNumCursors;
318. DWORD dwReserved;
319. [size\_is(cNumCursors)] DS\_REPL\_CURSOR rgCursor[];
320. } DS\_REPL\_CURSORS;
321. typedef struct {
322. [string] LPWSTR pszAttributeName;
323. DWORD dwVersion;
324. FILETIME ftimeLastOriginatingChange;
325. UUID uuidLastOriginatingDsaInvocationID;
326. USN usnOriginatingChange;
327. USN usnLocalChange;
328. } DS\_REPL\_ATTR\_META\_DATA;
329. typedef struct {
330. [string] LPWSTR pszDsaDN;
331. UUID uuidDsaObjGuid;
332. FILETIME ftimeFirstFailure;
333. DWORD cNumFailures;
334. DWORD dwLastResult;
335. } DS\_REPL\_KCC\_DSA\_FAILUREW;
336. typedef struct {
337. DWORD cNumEntries;
338. DWORD dwReserved;
339. [size\_is(cNumEntries)] DS\_REPL\_KCC\_DSA\_FAILUREW rgDsaFailure[];
340. } DS\_REPL\_KCC\_DSA\_FAILURESW;
341. typedef struct {
342. DWORD cNumEntries;
343. DWORD dwReserved;
344. [size\_is(cNumEntries)] DS\_REPL\_ATTR\_META\_DATA rgMetaData[];
345. } DS\_REPL\_OBJ\_META\_DATA;
346. typedef enum {
347. DS\_REPL\_OP\_TYPE\_SYNC = 0,
348. DS\_REPL\_OP\_TYPE\_ADD,
349. DS\_REPL\_OP\_TYPE\_DELETE,
350. DS\_REPL\_OP\_TYPE\_MODIFY,
351. DS\_REPL\_OP\_TYPE\_UPDATE\_REFS
352. } DS\_REPL\_OP\_TYPE;
353. typedef struct {
354. FILETIME ftimeEnqueued;
355. ULONG ulSerialNumber;
356. ULONG ulPriority;
357. DS\_REPL\_OP\_TYPE OpType;
358. ULONG ulOptions;
359. [string] LPWSTR pszNamingContext;
360. [string] LPWSTR pszDsaDN;
361. [string] LPWSTR pszDsaAddress;
362. UUID uuidNamingContextObjGuid;
363. UUID uuidDsaObjGuid;
364. } DS\_REPL\_OPW;
365. typedef struct {
366. FILETIME ftimeCurrentOpStarted;
367. DWORD cNumPendingOps;
368. [size\_is(cNumPendingOps)] DS\_REPL\_OPW rgPendingOp[];
369. } DS\_REPL\_PENDING\_OPSW;
370. typedef struct {
371. [string] LPWSTR pszAttributeName;
372. [string] LPWSTR pszObjectDn;
373. DWORD cbData;
374. [size\_is(cbData), ptr] BYTE \*pbData;
375. FILETIME ftimeDeleted;
376. FILETIME ftimeCreated;
377. DWORD dwVersion;
378. FILETIME ftimeLastOriginatingChange;
379. UUID uuidLastOriginatingDsaInvocationID;
380. USN usnOriginatingChange;
381. USN usnLocalChange;
382. } DS\_REPL\_VALUE\_META\_DATA;
383. typedef struct {
384. DWORD cNumEntries;
385. DWORD dwEnumerationContext;
386. [size\_is(cNumEntries)] DS\_REPL\_VALUE\_META\_DATA rgMetaData[];
387. } DS\_REPL\_ATTR\_VALUE\_META\_DATA;
388. typedef struct {
389. UUID uuidSourceDsaInvocationID;
390. USN usnAttributeFilter;
391. FILETIME ftimeLastSyncSuccess;
392. } DS\_REPL\_CURSOR\_2;
393. typedef struct {
394. DWORD cNumCursors;
395. DWORD dwEnumerationContext;
396. [size\_is(cNumCursors)] DS\_REPL\_CURSOR\_2 rgCursor[];
397. } DS\_REPL\_CURSORS\_2;
398. typedef struct {
399. UUID uuidSourceDsaInvocationID;
400. USN usnAttributeFilter;
401. FILETIME ftimeLastSyncSuccess;
402. [string] LPWSTR pszSourceDsaDN;
403. } DS\_REPL\_CURSOR\_3W;
404. typedef struct {
405. DWORD cNumCursors;
406. DWORD dwEnumerationContext;
407. [size\_is(cNumCursors)] DS\_REPL\_CURSOR\_3W rgCursor[];
408. } DS\_REPL\_CURSORS\_3W;
409. typedef struct {
410. [string] LPWSTR pszAttributeName;
411. DWORD dwVersion;
412. FILETIME ftimeLastOriginatingChange;
413. UUID uuidLastOriginatingDsaInvocationID;
414. USN usnOriginatingChange;
415. USN usnLocalChange;
416. [string] LPWSTR pszLastOriginatingDsaDN;
417. } DS\_REPL\_ATTR\_META\_DATA\_2;
418. typedef struct {
419. DWORD cNumEntries;
420. DWORD dwReserved;
421. [size\_is(cNumEntries)] DS\_REPL\_ATTR\_META\_DATA\_2 rgMetaData[];
422. } DS\_REPL\_OBJ\_META\_DATA\_2;
423. typedef struct {
424. [string] LPWSTR pszAttributeName;
425. [string] LPWSTR pszObjectDn;
426. DWORD cbData;
427. [size\_is(cbData), ptr] BYTE \*pbData;
428. FILETIME ftimeDeleted;
429. FILETIME ftimeCreated;
430. DWORD dwVersion;
431. FILETIME ftimeLastOriginatingChange;
432. UUID uuidLastOriginatingDsaInvocationID;
433. USN usnOriginatingChange;
434. USN usnLocalChange;
435. [string] LPWSTR pszLastOriginatingDsaDN;
436. } DS\_REPL\_VALUE\_META\_DATA\_2;
437. typedef struct {
438. DWORD cNumEntries;
439. DWORD dwEnumerationContext;
440. [size\_is(cNumEntries)] DS\_REPL\_VALUE\_META\_DATA\_2 rgMetaData[];
441. } DS\_REPL\_ATTR\_VALUE\_META\_DATA\_2;
442. typedef struct {
443. [range(1,10000)] DWORD cb;
444. [size\_is(cb)] BYTE rgb[];
445. } DRS\_EXTENSIONS;
446. typedef struct {
447. UUID uuidDsaObjDest;
448. UUID uuidInvocIdSrc;
449. [ref] DSNAME \*pNC;
450. USN\_VECTOR usnvecFrom;
451. [unique] UPTODATE\_VECTOR\_V1\_EXT \*pUpToDateVecDestV1;
452. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT \*pPartialAttrVecDestV1;
453. SCHEMA\_PREFIX\_TABLE PrefixTableDest;
454. ULONG ulFlags;
455. ULONG cMaxObjects;
456. ULONG cMaxBytes;
457. ULONG ulExtendedOp;
458. } DRS\_MSG\_GETCHGREQ\_V3;
459. typedef struct {
460. UUID uuidTransportObj;
461. [ref] MTX\_ADDR \*pmtxReturnAddress;
462. DRS\_MSG\_GETCHGREQ\_V3 V3;
463. } DRS\_MSG\_GETCHGREQ\_V4;
464. typedef struct {
465. UUID uuidTransportObj;
466. [ref] MTX\_ADDR \*pmtxReturnAddress;
467. DRS\_MSG\_GETCHGREQ\_V3 V3;
468. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT \*pPartialAttrSet;
469. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT \*pPartialAttrSetEx;
470. SCHEMA\_PREFIX\_TABLE PrefixTableDest;
471. } DRS\_MSG\_GETCHGREQ\_V7;
472. typedef struct {
473. UUID uuidDsaObjSrc;
474. UUID uuidInvocIdSrc;
475. [unique] DSNAME \*pNC;
476. USN\_VECTOR usnvecFrom;
477. USN\_VECTOR usnvecTo;
478. [unique] UPTODATE\_VECTOR\_V1\_EXT \*pUpToDateVecSrcV1;
479. SCHEMA\_PREFIX\_TABLE PrefixTableSrc;
480. ULONG ulExtendedRet;
481. ULONG cNumObjects;
482. ULONG cNumBytes;
483. [unique] REPLENTINFLIST\* pObjects;
484. BOOL fMoreData;
485. } DRS\_MSG\_GETCHGREPLY\_V1;
486. typedef struct {
487. UUID uuidDsaObjSrc;
488. UUID uuidInvocIdSrc;
489. [unique] DSNAME \*pNC;
490. USN\_VECTOR usnvecFrom;
491. USN\_VECTOR usnvecTo;
492. [unique] UPTODATE\_VECTOR\_V2\_EXT \*pUpToDateVecSrc;
493. SCHEMA\_PREFIX\_TABLE PrefixTableSrc;
494. ULONG ulExtendedRet;
495. ULONG cNumObjects;
496. ULONG cNumBytes;
497. [unique] REPLENTINFLIST \*pObjects;
498. BOOL fMoreData;
499. ULONG cNumNcSizeObjects;
500. ULONG cNumNcSizeValues;
501. [range(0,1048576)] DWORD cNumValues;
502. [size\_is(cNumValues)] REPLVALINF\_V1 \*rgValues;
503. DWORD dwDRSError;
504. } DRS\_MSG\_GETCHGREPLY\_V6;
505. typedef struct {
506. UUID uuidDsaObjSrc;
507. UUID uuidInvocIdSrc;
508. [unique] DSNAME \*pNC;
509. USN\_VECTOR usnvecFrom;
510. USN\_VECTOR usnvecTo;
511. [unique] UPTODATE\_VECTOR\_V2\_EXT \*pUpToDateVecSrc;
512. SCHEMA\_PREFIX\_TABLE PrefixTableSrc;
513. ULONG ulExtendedRet;
514. ULONG cNumObjects;
515. ULONG cNumBytes;
516. [unique] REPLENTINFLIST \*pObjects;
517. BOOL fMoreData;
518. ULONG cNumNcSizeObjects;
519. ULONG cNumNcSizeValues;
520. [range(0,1048576)] DWORD cNumValues;
521. [size\_is(cNumValues)] REPLVALINF\_V3 \*rgValues;
522. DWORD dwDRSError;
523. } DRS\_MSG\_GETCHGREPLY\_V9;
524. typedef struct {
525. DWORD cbUncompressedSize;
526. DWORD cbCompressedSize;
527. [size\_is(cbCompressedSize)] BYTE \*pbCompressedData;
528. } DRS\_COMPRESSED\_BLOB;
529. typedef struct {
530. UUID uuidDsaObjDest;
531. UUID uuidInvocIdSrc;
532. [ref] DSNAME \*pNC;
533. USN\_VECTOR usnvecFrom;
534. [unique] UPTODATE\_VECTOR\_V1\_EXT \*pUpToDateVecDestV1;
535. ULONG ulFlags;
536. ULONG cMaxObjects;
537. ULONG cMaxBytes;
538. ULONG ulExtendedOp;
539. ULARGE\_INTEGER liFsmoInfo;
540. } DRS\_MSG\_GETCHGREQ\_V5;
541. typedef struct {
542. UUID uuidDsaObjDest;
543. UUID uuidInvocIdSrc;
544. [ref] DSNAME \*pNC;
545. USN\_VECTOR usnvecFrom;
546. [unique] UPTODATE\_VECTOR\_V1\_EXT \*pUpToDateVecDest;
547. ULONG ulFlags;
548. ULONG cMaxObjects;
549. ULONG cMaxBytes;
550. ULONG ulExtendedOp;
551. ULARGE\_INTEGER liFsmoInfo;
552. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT \*pPartialAttrSet;
553. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT \*pPartialAttrSetEx;
554. SCHEMA\_PREFIX\_TABLE PrefixTableDest;
555. } DRS\_MSG\_GETCHGREQ\_V8;
556. typedef struct {
557. UUID uuidDsaObjDest;
558. UUID uuidInvocIdSrc;
559. [ref] DSNAME \*pNC;
560. USN\_VECTOR usnvecFrom;
561. [unique] UPTODATE\_VECTOR\_V1\_EXT \*pUpToDateVecDest;
562. ULONG ulFlags;
563. ULONG cMaxObjects;
564. ULONG cMaxBytes;
565. ULONG ulExtendedOp;
566. ULARGE\_INTEGER liFsmoInfo;
567. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT \*pPartialAttrSet;
568. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT \*pPartialAttrSetEx;
569. SCHEMA\_PREFIX\_TABLE PrefixTableDest;
570. ULONG ulMoreFlags;
571. } DRS\_MSG\_GETCHGREQ\_V10;
572. typedef struct {
573. ULONG ulVersion;
574. ULONG cbByteBuffer;
575. ULONGLONG ullPadding;
576. [size\_is(cbByteBuffer)] BYTE rgbBuffer[];
577. } VAR\_SIZE\_BUFFER\_WITH\_VERSION;
578. typedef struct {
579. UUID uuidDsaObjDest;
580. UUID uuidInvocIdSrc;
581. [ref] DSNAME \*pNC;
582. USN\_VECTOR usnvecFrom;
583. [unique] UPTODATE\_VECTOR\_V1\_EXT \*pUpToDateVecDest;
584. ULONG ulFlags;
585. ULONG cMaxObjects;
586. ULONG cMaxBytes;
587. ULONG ulExtendedOp;
588. ULARGE\_INTEGER liFsmoInfo;
589. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT \*pPartialAttrSet;
590. [unique] PARTIAL\_ATTR\_VECTOR\_V1\_EXT \*pPartialAttrSetEx;
591. SCHEMA\_PREFIX\_TABLE PrefixTableDest;
592. ULONG ulMoreFlags;
593. GUID correlationID;
594. [unique] VAR\_SIZE\_BUFFER\_WITH\_VERSION \*pReservedBuffer;
595. } DRS\_MSG\_GETCHGREQ\_V11;
596. typedef [switch\_type(DWORD)] union {
597. [case(4)] DRS\_MSG\_GETCHGREQ\_V4 V4;
598. [case(5)] DRS\_MSG\_GETCHGREQ\_V5 V5;
599. [case(7)] DRS\_MSG\_GETCHGREQ\_V7 V7;
600. [case(8)] DRS\_MSG\_GETCHGREQ\_V8 V8;
601. [case(10)] DRS\_MSG\_GETCHGREQ\_V10 V10;
602. [case(11)] DRS\_MSG\_GETCHGREQ\_V11 V11;
603. } DRS\_MSG\_GETCHGREQ;
604. typedef struct {
605. DRS\_COMPRESSED\_BLOB CompressedV1;
606. } DRS\_MSG\_GETCHGREPLY\_V2;
607. typedef enum {
608. DRS\_COMP\_ALG\_NONE = 0,
609. DRS\_COMP\_ALG\_UNUSED = 1,
610. DRS\_COMP\_ALG\_MSZIP = 2,
611. DRS\_COMP\_ALG\_WIN2K3 = 3
612. } DRS\_COMP\_ALG\_TYPE;
613. typedef struct {
614. DWORD dwCompressedVersion;
615. DRS\_COMP\_ALG\_TYPE CompressionAlg;
616. DRS\_COMPRESSED\_BLOB CompressedAny;
617. } DRS\_MSG\_GETCHGREPLY\_V7;
618. typedef [switch\_type(DWORD)] union {
619. [case(1)] DRS\_MSG\_GETCHGREPLY\_V1 V1;
620. [case(2)] DRS\_MSG\_GETCHGREPLY\_V2 V2;
621. [case(6)] DRS\_MSG\_GETCHGREPLY\_V6 V6;
622. [case(7)] DRS\_MSG\_GETCHGREPLY\_V7 V7;
623. [case(9)] DRS\_MSG\_GETCHGREPLY\_V9 V9;
624. } DRS\_MSG\_GETCHGREPLY;
625. typedef struct {
626. [ref] DSNAME \*pNC;
627. UUID uuidDsaSrc;
628. [unique] [string] char \*pszDsaSrc;
629. ULONG ulOptions;
630. } DRS\_MSG\_REPSYNC\_V1;
631. typedef struct {
632. [ref] DSNAME \*pNC;
633. UUID uuidDsaSrc;
634. [unique] [string] char \*pszDsaSrc;
635. ULONG ulOptions;
636. GUID correlationID;
637. [unique] VAR\_SIZE\_BUFFER\_WITH\_VERSION \*pReservedBuffer;
638. } DRS\_MSG\_REPSYNC\_V2;
639. typedef [switch\_type(DWORD)] union
640. {
641. [case(1)] DRS\_MSG\_REPSYNC\_V1 V1;
642. [case(2)] DRS\_MSG\_REPSYNC\_V2 V2;
643. } DRS\_MSG\_REPSYNC;
644. typedef struct {
645. [ref] DSNAME \*pNC;
646. [ref] [string] char \*pszDsaDest;
647. UUID uuidDsaObjDest;
648. ULONG ulOptions;
649. } DRS\_MSG\_UPDREFS\_V1;
650. typedef struct {
651. [ref] DSNAME \*pNC;
652. [ref] [string] char \*pszDsaDest;
653. UUID uuidDsaObjDest;
654. ULONG ulOptions;
655. GUID correlationID;
656. [unique] VAR\_SIZE\_BUFFER\_WITH\_VERSION\* pReservedBuffer;
657. } DRS\_MSG\_UPDREFS\_V2;
658. typedef [switch\_type(DWORD)] union {
659. [case(1)] DRS\_MSG\_UPDREFS\_V1 V1;
660. [case(2)] DRS\_MSG\_UPDREFS\_V2 V2;
661. } DRS\_MSG\_UPDREFS;
662. typedef struct {
663. [ref] DSNAME \*pNC;
664. [ref] [string] char \*pszDsaSrc;
665. REPLTIMES rtSchedule;
666. ULONG ulOptions;
667. } DRS\_MSG\_REPADD\_V1;
668. typedef struct {
669. [ref] DSNAME \*pNC;
670. [unique] DSNAME \*pSourceDsaDN;
671. [unique] DSNAME \*pTransportDN;
672. [ref] [string] char \*pszSourceDsaAddress;
673. REPLTIMES rtSchedule;
674. ULONG ulOptions;
675. } DRS\_MSG\_REPADD\_V2;
676. typedef struct {
677. [ref] DSNAME \*pNC;
678. [unique] DSNAME \*pSourceDsaDN;
679. [unique] DSNAME \*pTransportDN;
680. [ref] [string] char \*pszSourceDsaAddress;
681. REPLTIMES rtSchedule;
682. ULONG ulOptions;
683. GUID correlationID;
684. [unique] VAR\_SIZE\_BUFFER\_WITH\_VERSION\* pReservedBuffer;
685. } DRS\_MSG\_REPADD\_V3;
686. typedef [switch\_type(DWORD)] union {
687. [case(1)] DRS\_MSG\_REPADD\_V1 V1;
688. [case(2)] DRS\_MSG\_REPADD\_V2 V2;
689. [case(3)] DRS\_MSG\_REPADD\_V3 V3;
690. } DRS\_MSG\_REPADD;
691. typedef struct {
692. [ref] DSNAME \*pNC;
693. [string] char \*pszDsaSrc;
694. ULONG ulOptions;
695. } DRS\_MSG\_REPDEL\_V1;
696. typedef [switch\_type(DWORD)] union {
697. [case(1)] DRS\_MSG\_REPDEL\_V1 V1;
698. } DRS\_MSG\_REPDEL;
699. typedef struct {
700. [ref] DSNAME \*pNC;
701. UUID uuidSourceDRA;
702. [unique, string] char \*pszSourceDRA;
703. REPLTIMES rtSchedule;
704. ULONG ulReplicaFlags;
705. ULONG ulModifyFields;
706. ULONG ulOptions;
707. } DRS\_MSG\_REPMOD\_V1;
708. typedef [switch\_type(DWORD)] union {
709. [case(1)] DRS\_MSG\_REPMOD\_V1 V1;
710. } DRS\_MSG\_REPMOD;
711. typedef struct {
712. DWORD dwFlags;
713. [range(1,10000)] DWORD cNames;
714. [size\_is(cNames)] DSNAME \*\*rpNames;
715. ATTRBLOCK RequiredAttrs;
716. SCHEMA\_PREFIX\_TABLE PrefixTable;
717. } DRS\_MSG\_VERIFYREQ\_V1;
718. typedef [switch\_type(DWORD)] union {
719. [case(1)] DRS\_MSG\_VERIFYREQ\_V1 V1;
720. } DRS\_MSG\_VERIFYREQ;
721. typedef struct {
722. DWORD error;
723. [range(0,10000)] DWORD cNames;
724. [size\_is(cNames)] ENTINF \*rpEntInf;
725. SCHEMA\_PREFIX\_TABLE PrefixTable;
726. } DRS\_MSG\_VERIFYREPLY\_V1;
727. typedef [switch\_type(DWORD)] union {
728. [case(1)] DRS\_MSG\_VERIFYREPLY\_V1 V1;
729. } DRS\_MSG\_VERIFYREPLY;
730. typedef enum {
731. RevMembGetGroupsForUser=1,
732. RevMembGetAliasMembership,
733. RevMembGetAccountGroups,
734. RevMembGetResourceGroups,
735. RevMembGetUniversalGroups,
736. GroupMembersTransitive,
737. RevMembGlobalGroupsNonTransitive
738. } REVERSE\_MEMBERSHIP\_OPERATION\_TYPE;
739. typedef struct {
740. [range(1,10000)] ULONG cDsNames;
741. [size\_is(cDsNames,)] DSNAME \*\*ppDsNames;
742. DWORD dwFlags;
743. [range(1,7)] REVERSE\_MEMBERSHIP\_OPERATION\_TYPE OperationType;
744. DSNAME \*pLimitingDomain;
745. } DRS\_MSG\_REVMEMB\_REQ\_V1;
746. typedef [switch\_type(DWORD)] union {
747. [case(1)] DRS\_MSG\_REVMEMB\_REQ\_V1 V1;
748. } DRS\_MSG\_REVMEMB\_REQ;
749. typedef struct {
750. ULONG errCode;
751. [range(0,10000)] ULONG cDsNames;
752. [range(0,10000)] ULONG cSidHistory;
753. [size\_is(cDsNames,)] DSNAME \*\*ppDsNames;
754. [size\_is(cDsNames)] DWORD \*pAttributes;
755. [size\_is(cSidHistory,)] NT4SID \*\*ppSidHistory;
756. } DRS\_MSG\_REVMEMB\_REPLY\_V1;
757. typedef [switch\_type(DWORD)] union {
758. [case(1)] DRS\_MSG\_REVMEMB\_REPLY\_V1 V1;
759. } DRS\_MSG\_REVMEMB\_REPLY;
760. typedef struct {
761. char \*pSourceDSA;
762. ENTINF \*pObject;
763. UUID \*pParentUUID;
764. SCHEMA\_PREFIX\_TABLE PrefixTable;
765. ULONG ulFlags;
766. } DRS\_MSG\_MOVEREQ\_V1;
767. typedef struct {
768. [range(0,10000)] unsigned long cbBuffer;
769. unsigned long BufferType;
770. [size\_is(cbBuffer)] BYTE \*pvBuffer;
771. } DRS\_SecBuffer;
772. typedef struct {
773. unsigned long ulVersion;
774. [range(0,10000)] unsigned long cBuffers;
775. [size\_is(cBuffers)] DRS\_SecBuffer \*Buffers;
776. } DRS\_SecBufferDesc;
777. typedef struct {
778. DSNAME \*pSrcDSA;
779. ENTINF \*pSrcObject;
780. DSNAME \*pDstName;
781. DSNAME \*pExpectedTargetNC;
782. DRS\_SecBufferDesc \*pClientCreds;
783. SCHEMA\_PREFIX\_TABLE PrefixTable;
784. ULONG ulFlags;
785. } DRS\_MSG\_MOVEREQ\_V2;
786. typedef [switch\_type(DWORD)] union {
787. [case(1)] DRS\_MSG\_MOVEREQ\_V1 V1;
788. [case(2)] DRS\_MSG\_MOVEREQ\_V2 V2;
789. } DRS\_MSG\_MOVEREQ;
790. typedef struct {
791. ENTINF \*\*ppResult;
792. SCHEMA\_PREFIX\_TABLE PrefixTable;
793. ULONG \*pError;
794. } DRS\_MSG\_MOVEREPLY\_V1;
795. typedef struct {
796. ULONG win32Error;
797. [unique] DSNAME \*pAddedName;
798. } DRS\_MSG\_MOVEREPLY\_V2;
799. typedef [switch\_type(DWORD)] union {
800. [case(1)] DRS\_MSG\_MOVEREPLY\_V1 V1;
801. [case(2)] DRS\_MSG\_MOVEREPLY\_V2 V2;
802. } DRS\_MSG\_MOVEREPLY;
803. typedef struct {
804. ULONG CodePage;
805. ULONG LocaleId;
806. DWORD dwFlags;
807. DWORD formatOffered;
808. DWORD formatDesired;
809. [range(1,10000)] DWORD cNames;
810. [string, size\_is(cNames)] WCHAR \*\*rpNames;
811. } DRS\_MSG\_CRACKREQ\_V1;
812. typedef [switch\_type(DWORD)] union {
813. [case(1)] DRS\_MSG\_CRACKREQ\_V1 V1;
814. } DRS\_MSG\_CRACKREQ;
815. typedef struct {
816. DS\_NAME\_RESULTW \*pResult;
817. } DRS\_MSG\_CRACKREPLY\_V1;
818. typedef [switch\_type(DWORD)] union {
819. [case(1)] DRS\_MSG\_CRACKREPLY\_V1 V1;
820. } DRS\_MSG\_CRACKREPLY;
821. typedef struct {
822. DWORD dwFlags;
823. DWORD PreferredMaximumLength;
824. [range(0,10485760)] DWORD cbRestart;
825. [size\_is(cbRestart)] BYTE \*pRestart;
826. } DRS\_MSG\_NT4\_CHGLOG\_REQ\_V1;
827. typedef [switch\_type(DWORD)] union {
828. [case(1)] DRS\_MSG\_NT4\_CHGLOG\_REQ\_V1 V1;
829. } DRS\_MSG\_NT4\_CHGLOG\_REQ;
830. typedef struct {
831. LARGE\_INTEGER SamSerialNumber;
832. LARGE\_INTEGER SamCreationTime;
833. LARGE\_INTEGER BuiltinSerialNumber;
834. LARGE\_INTEGER BuiltinCreationTime;
835. LARGE\_INTEGER LsaSerialNumber;
836. LARGE\_INTEGER LsaCreationTime;
837. } NT4\_REPLICATION\_STATE;
838. typedef struct {
839. [range(0,10485760)] DWORD cbRestart;
840. [range(0,10485760)] DWORD cbLog;
841. NT4\_REPLICATION\_STATE ReplicationState;
842. DWORD ActualNtStatus;
843. [size\_is(cbRestart)] BYTE \*pRestart;
844. [size\_is(cbLog)] BYTE \*pLog;
845. } DRS\_MSG\_NT4\_CHGLOG\_REPLY\_V1;
846. typedef [switch\_type(DWORD)] union {
847. [case(1)] DRS\_MSG\_NT4\_CHGLOG\_REPLY\_V1 V1;
848. } DRS\_MSG\_NT4\_CHGLOG\_REPLY;
849. typedef struct {
850. DWORD operation;
851. DWORD flags;
852. [string] const WCHAR \*pwszAccount;
853. [range(0,10000)] DWORD cSPN;
854. [string, size\_is(cSPN)] const WCHAR \*\*rpwszSPN;
855. } DRS\_MSG\_SPNREQ\_V1;
856. typedef [switch\_type(DWORD)] union {
857. [case(1)] DRS\_MSG\_SPNREQ\_V1 V1;
858. } DRS\_MSG\_SPNREQ;
859. typedef struct {
860. DWORD retVal;
861. } DRS\_MSG\_SPNREPLY\_V1;
862. typedef [switch\_type(DWORD)] union {
863. [case(1)] DRS\_MSG\_SPNREPLY\_V1 V1;
864. } DRS\_MSG\_SPNREPLY;
865. typedef struct {
866. [string] LPWSTR ServerDN;
867. [string] LPWSTR DomainDN;
868. BOOL fCommit;
869. } DRS\_MSG\_RMSVRREQ\_V1;
870. typedef [switch\_type(DWORD)] union {
871. [case(1)] DRS\_MSG\_RMSVRREQ\_V1 V1;
872. } DRS\_MSG\_RMSVRREQ;
873. typedef struct {
874. BOOL fLastDcInDomain;
875. } DRS\_MSG\_RMSVRREPLY\_V1;
876. typedef [switch\_type(DWORD)] union {
877. [case(1)] DRS\_MSG\_RMSVRREPLY\_V1 V1;
878. } DRS\_MSG\_RMSVRREPLY;
879. typedef struct {
880. [string] LPWSTR DomainDN;
881. } DRS\_MSG\_RMDMNREQ\_V1;
882. typedef [switch\_type(DWORD)] union {
883. [case(1)] DRS\_MSG\_RMDMNREQ\_V1 V1;
884. } DRS\_MSG\_RMDMNREQ;
885. typedef struct {
886. DWORD Reserved;
887. } DRS\_MSG\_RMDMNREPLY\_V1;
888. typedef [switch\_type(DWORD)] union {
889. [case(1)] DRS\_MSG\_RMDMNREPLY\_V1 V1;
890. } DRS\_MSG\_RMDMNREPLY;
891. typedef struct {
892. [string] WCHAR \*Domain;
893. DWORD InfoLevel;
894. } DRS\_MSG\_DCINFOREQ\_V1;
895. typedef [switch\_type(DWORD)] union {
896. [case(1)] DRS\_MSG\_DCINFOREQ\_V1 V1;
897. } DRS\_MSG\_DCINFOREQ, \*PDRS\_MSG\_DCINFOREQ;
898. typedef struct {
899. [range(0,10000)] DWORD cItems;
900. [size\_is(cItems)] DS\_DOMAIN\_CONTROLLER\_INFO\_1W \*rItems;
901. } DRS\_MSG\_DCINFOREPLY\_V1;
902. typedef struct {
903. [range(0,10000)] DWORD cItems;
904. [size\_is(cItems)] DS\_DOMAIN\_CONTROLLER\_INFO\_2W \*rItems;
905. } DRS\_MSG\_DCINFOREPLY\_V2;
906. typedef struct {
907. [range(0,10000)] DWORD cItems;
908. [size\_is(cItems)] DS\_DOMAIN\_CONTROLLER\_INFO\_3W\* rItems;
909. } DRS\_MSG\_DCINFOREPLY\_V3;
910. typedef struct {
911. [range(0,10000)] DWORD cItems;
912. [size\_is(cItems)] DS\_DOMAIN\_CONTROLLER\_INFO\_FFFFFFFFW \*rItems;
913. } DRS\_MSG\_DCINFOREPLY\_VFFFFFFFF;
914. typedef [switch\_type(DWORD)] union {
915. [case(1)] DRS\_MSG\_DCINFOREPLY\_V1 V1;
916. [case(2)] DRS\_MSG\_DCINFOREPLY\_V2 V2;
917. [case(3)] DRS\_MSG\_DCINFOREPLY\_V3 V3;
918. [case(0xFFFFFFFF)] DRS\_MSG\_DCINFOREPLY\_VFFFFFFFF VFFFFFFFF;
919. } DRS\_MSG\_DCINFOREPLY;
920. typedef struct {
921. [ref] DSNAME \*pObject;
922. ATTRBLOCK AttrBlock;
923. } DRS\_MSG\_ADDENTRYREQ\_V1;
924. typedef struct {
925. ENTINFLIST EntInfList;
926. } DRS\_MSG\_ADDENTRYREQ\_V2;
927. typedef struct {
928. ENTINFLIST EntInfList;
929. DRS\_SecBufferDesc \*pClientCreds;
930. } DRS\_MSG\_ADDENTRYREQ\_V3;
931. typedef [switch\_type(DWORD)] union {
932. [case(1)] DRS\_MSG\_ADDENTRYREQ\_V1 V1;
933. [case(2)] DRS\_MSG\_ADDENTRYREQ\_V2 V2;
934. [case(3)] DRS\_MSG\_ADDENTRYREQ\_V3 V3;
935. } DRS\_MSG\_ADDENTRYREQ;
936. typedef struct {
937. GUID Guid;
938. NT4SID Sid;
939. DWORD errCode;
940. DWORD dsid;
941. DWORD extendedErr;
942. DWORD extendedData;
943. USHORT problem;
944. } DRS\_MSG\_ADDENTRYREPLY\_V1;
945. typedef struct {
946. GUID objGuid;
947. NT4SID objSid;
948. } ADDENTRY\_REPLY\_INFO;
949. typedef struct {
950. [unique] DSNAME \*pErrorObject;
951. DWORD errCode;
952. DWORD dsid;
953. DWORD extendedErr;
954. DWORD extendedData;
955. USHORT problem;
956. [range(0,10000)] ULONG cObjectsAdded;
957. [size\_is(cObjectsAdded)] ADDENTRY\_REPLY\_INFO \*infoList;
958. } DRS\_MSG\_ADDENTRYREPLY\_V2;
959. typedef struct {
960. DWORD dwRepError;
961. DWORD errCode;
962. [switch\_is(errCode)] DIRERR\_DRS\_WIRE\_V1 \*pErrInfo;
963. } DRS\_ERROR\_DATA\_V1;
964. typedef [switch\_type(DWORD)] union {
965. [case(1)] DRS\_ERROR\_DATA\_V1 V1;
966. } DRS\_ERROR\_DATA;
967. typedef struct {
968. DSNAME \*pdsErrObject;
969. DWORD dwErrVer;
970. [switch\_is(dwErrVer)] DRS\_ERROR\_DATA \*pErrData;
971. [range(0,10000)] ULONG cObjectsAdded;
972. [size\_is(cObjectsAdded)] ADDENTRY\_REPLY\_INFO \*infoList;
973. } DRS\_MSG\_ADDENTRYREPLY\_V3;
974. typedef [switch\_type(DWORD)] union {
975. [case(1)] DRS\_MSG\_ADDENTRYREPLY\_V1 V1;
976. [case(2)] DRS\_MSG\_ADDENTRYREPLY\_V2 V2;
977. [case(3)] DRS\_MSG\_ADDENTRYREPLY\_V3 V3;
978. } DRS\_MSG\_ADDENTRYREPLY;
979. typedef struct {
980. DWORD dwTaskID;
981. DWORD dwFlags;
982. } DRS\_MSG\_KCC\_EXECUTE\_V1;
983. typedef [switch\_type(DWORD)] union {
984. [case(1)] DRS\_MSG\_KCC\_EXECUTE\_V1 V1;
985. } DRS\_MSG\_KCC\_EXECUTE;
986. typedef struct {
987. ULONGLONG hCtx;
988. LONG lReferenceCount;
989. BOOL fIsBound;
990. UUID uuidClient;
991. DSTIME timeLastUsed;
992. ULONG IPAddr;
993. int pid;
994. } DS\_REPL\_CLIENT\_CONTEXT;
995. typedef struct {
996. [range(0,10000)] DWORD cNumContexts;
997. DWORD dwReserved;
998. [size\_is(cNumContexts)] DS\_REPL\_CLIENT\_CONTEXT rgContext[];
999. } DS\_REPL\_CLIENT\_CONTEXTS;
1000. typedef struct {
1001. [string] LPWSTR pszServerName;
1002. BOOL fIsHandleBound;
1003. BOOL fIsHandleFromCache;
1004. BOOL fIsHandleInCache;
1005. DWORD dwThreadId;
1006. DWORD dwBindingTimeoutMins;
1007. DSTIME dstimeCreated;
1008. DWORD dwCallType;
1009. } DS\_REPL\_SERVER\_OUTGOING\_CALL;
1010. typedef struct {
1011. [range(0, 256)] DWORD cNumCalls;
1012. DWORD dwReserved;
1013. [size\_is(cNumCalls)] DS\_REPL\_SERVER\_OUTGOING\_CALL rgCall[];
1014. } DS\_REPL\_SERVER\_OUTGOING\_CALLS;
1015. typedef struct {
1016. DWORD InfoType;
1017. [string] LPWSTR pszObjectDN;
1018. UUID uuidSourceDsaObjGuid;
1019. } DRS\_MSG\_GETREPLINFO\_REQ\_V1;
1020. typedef struct {
1021. DWORD InfoType;
1022. [string] LPWSTR pszObjectDN;
1023. UUID uuidSourceDsaObjGuid;
1024. DWORD ulFlags;
1025. [string] LPWSTR pszAttributeName;
1026. [string] LPWSTR pszValueDN;
1027. DWORD dwEnumerationContext;
1028. } DRS\_MSG\_GETREPLINFO\_REQ\_V2;
1029. typedef [switch\_type(DWORD)] union {
1030. [case(1)] DRS\_MSG\_GETREPLINFO\_REQ\_V1 V1;
1031. [case(2)] DRS\_MSG\_GETREPLINFO\_REQ\_V2 V2;
1032. } DRS\_MSG\_GETREPLINFO\_REQ;
1033. typedef [switch\_type(DWORD)] union {
1034. [case(0)] DS\_REPL\_NEIGHBORSW \*pNeighbors;
1035. [case(1)] DS\_REPL\_CURSORS \*pCursors;
1036. [case(2)] DS\_REPL\_OBJ\_META\_DATA \*pObjMetaData;
1037. [case(3)] DS\_REPL\_KCC\_DSA\_FAILURESW \*pConnectFailures;
1038. [case(4)] DS\_REPL\_KCC\_DSA\_FAILURESW \*pLinkFailures;
1039. [case(5)] DS\_REPL\_PENDING\_OPSW \*pPendingOps;
1040. [case(6)] DS\_REPL\_ATTR\_VALUE\_META\_DATA \*pAttrValueMetaData;
1041. [case(7)] DS\_REPL\_CURSORS\_2 \*pCursors2;
1042. [case(8)] DS\_REPL\_CURSORS\_3W \*pCursors3;
1043. [case(9)] DS\_REPL\_OBJ\_META\_DATA\_2 \*pObjMetaData2;
1044. [case(10)] DS\_REPL\_ATTR\_VALUE\_META\_DATA\_2 \*pAttrValueMetaData2;
1045. [case(0xFFFFFFFA)]
1046. DS\_REPL\_SERVER\_OUTGOING\_CALLS \*pServerOutgoingCalls;
1047. [case(0xFFFFFFFB)] UPTODATE\_VECTOR\_V1\_EXT \*pUpToDateVec;
1048. [case(0xFFFFFFFC)] DS\_REPL\_CLIENT\_CONTEXTS \*pClientContexts;
1049. [case(0xFFFFFFFE)] DS\_REPL\_NEIGHBORSW \*pRepsTo;
1050. } DRS\_MSG\_GETREPLINFO\_REPLY;
1051. typedef struct {
1052. DWORD Flags;
1053. [string] WCHAR \*SrcDomain;
1054. [string] WCHAR \*SrcPrincipal;
1055. [string, ptr] WCHAR \*SrcDomainController;
1056. [range(0,256)] DWORD SrcCredsUserLength;
1057. [size\_is(SrcCredsUserLength)] WCHAR \*SrcCredsUser;
1058. [range(0,256)] DWORD SrcCredsDomainLength;
1059. [size\_is(SrcCredsDomainLength)] WCHAR \*SrcCredsDomain;
1060. [range(0,256)] DWORD SrcCredsPasswordLength;
1061. [size\_is(SrcCredsPasswordLength)] WCHAR \*SrcCredsPassword;
1062. [string] WCHAR \*DstDomain;
1063. [string] WCHAR \*DstPrincipal;
1064. } DRS\_MSG\_ADDSIDREQ\_V1;
1065. typedef [switch\_type(DWORD)] union {
1066. [case(1)] DRS\_MSG\_ADDSIDREQ\_V1 V1;
1067. } DRS\_MSG\_ADDSIDREQ;
1068. typedef struct {
1069. DWORD dwWin32Error;
1070. } DRS\_MSG\_ADDSIDREPLY\_V1;
1071. typedef [switch\_type(DWORD)] union {
1072. [case(1)] DRS\_MSG\_ADDSIDREPLY\_V1 V1;
1073. } DRS\_MSG\_ADDSIDREPLY;
1074. typedef struct {
1075. [range(1, 10000)] ULONG Count;
1076. [size\_is(Count)] DRS\_MSG\_REVMEMB\_REQ\_V1 \*Requests;
1077. } DRS\_MSG\_GETMEMBERSHIPS2\_REQ\_V1;
1078. typedef [switch\_type(DWORD)] union {
1079. [case(1)] DRS\_MSG\_GETMEMBERSHIPS2\_REQ\_V1 V1;
1080. } DRS\_MSG\_GETMEMBERSHIPS2\_REQ;
1081. typedef struct {
1082. [range(0, 10000)] ULONG Count;
1083. [size\_is(Count)] DRS\_MSG\_REVMEMB\_REPLY\_V1 \*Replies;
1084. } DRS\_MSG\_GETMEMBERSHIPS2\_REPLY\_V1;
1085. typedef [switch\_type(DWORD)] union {
1086. [case(1)] DRS\_MSG\_GETMEMBERSHIPS2\_REPLY\_V1 V1;
1087. } DRS\_MSG\_GETMEMBERSHIPS2\_REPLY;
1088. typedef struct {
1089. [ref] DSNAME \*pNC;
1090. UUID uuidDsaSrc;
1091. ULONG ulOptions;
1092. } DRS\_MSG\_REPVERIFYOBJ\_V1;
1093. typedef [switch\_type(DWORD)] union {
1094. [case(1)] DRS\_MSG\_REPVERIFYOBJ\_V1 V1;
1095. } DRS\_MSG\_REPVERIFYOBJ;
1096. typedef struct {
1097. UUID guidStart;
1098. DWORD cGuids;
1099. DSNAME \*pNC;
1100. UPTODATE\_VECTOR\_V1\_EXT \*pUpToDateVecCommonV1;
1101. UCHAR Md5Digest[16];
1102. } DRS\_MSG\_EXISTREQ\_V1;
1103. typedef [switch\_type(DWORD)] union {
1104. [case(1)] DRS\_MSG\_EXISTREQ\_V1 V1;
1105. } DRS\_MSG\_EXISTREQ;
1106. typedef struct {
1107. DWORD dwStatusFlags;
1108. [range(0,10485760)] DWORD cNumGuids;
1109. [size\_is(cNumGuids)] UUID \*rgGuids;
1110. } DRS\_MSG\_EXISTREPLY\_V1;
1111. typedef [switch\_type(DWORD)] union {
1112. [case(1)] DRS\_MSG\_EXISTREPLY\_V1 V1;
1113. } DRS\_MSG\_EXISTREPLY;
1114. typedef struct {
1115. [string] const WCHAR \*pwszFromSite;
1116. [range(1,10000)] DWORD cToSites;
1117. [string, size\_is(cToSites)] WCHAR \*\*rgszToSites;
1118. DWORD dwFlags;
1119. } DRS\_MSG\_QUERYSITESREQ\_V1;
1120. typedef [switch\_type(DWORD)] union {
1121. [case(1)] DRS\_MSG\_QUERYSITESREQ\_V1 V1;
1122. } DRS\_MSG\_QUERYSITESREQ;
1123. typedef struct {
1124. DWORD dwErrorCode;
1125. DWORD dwCost;
1126. } DRS\_MSG\_QUERYSITESREPLYELEMENT\_V1;
1127. typedef struct {
1128. [range(0,10000)] DWORD cToSites;
1129. [size\_is(cToSites)] DRS\_MSG\_QUERYSITESREPLYELEMENT\_V1 \*rgCostInfo;
1130. DWORD dwFlags;
1131. } DRS\_MSG\_QUERYSITESREPLY\_V1;
1132. typedef [switch\_type(DWORD)] union {
1133. [case(1)] DRS\_MSG\_QUERYSITESREPLY\_V1 V1;
1134. } DRS\_MSG\_QUERYSITESREPLY;
1135. typedef struct {
1136. DWORD dwReserved;
1137. } DRS\_MSG\_INIT\_DEMOTIONREQ\_V1;
1138. typedef [switch\_type(DWORD)] union {
1139. [case(1)] DRS\_MSG\_INIT\_DEMOTIONREQ\_V1 V1;
1140. } DRS\_MSG\_INIT\_DEMOTIONREQ;
1141. typedef struct {
1142. DWORD dwOpError;
1143. } DRS\_MSG\_INIT\_DEMOTIONREPLY\_V1;
1144. typedef [switch\_type(DWORD)] union {
1145. [case(1)] DRS\_MSG\_INIT\_DEMOTIONREPLY\_V1 V1;
1146. } DRS\_MSG\_INIT\_DEMOTIONREPLY;
1147. typedef struct {
1148. DWORD dwFlags;
1149. UUID uuidHelperDest;
1150. [ref] DSNAME\* pNC;
1151. } DRS\_MSG\_REPLICA\_DEMOTIONREQ\_V1;
1152. typedef [switch\_type(DWORD)] union {
1153. [case(1)] DRS\_MSG\_REPLICA\_DEMOTIONREQ\_V1 V1;
1154. } DRS\_MSG\_REPLICA\_DEMOTIONREQ;
1155. typedef struct {
1156. DWORD dwOpError;
1157. } DRS\_MSG\_REPLICA\_DEMOTIONREPLY\_V1;
1158. typedef [switch\_type(DWORD)] union {
1159. [case(1)] DRS\_MSG\_REPLICA\_DEMOTIONREPLY\_V1 V1;
1160. } DRS\_MSG\_REPLICA\_DEMOTIONREPLY;
1161. typedef struct {
1162. DWORD dwOperations;
1163. UUID uuidHelperDest;
1164. [string] LPWSTR szScriptBase;
1165. } DRS\_MSG\_FINISH\_DEMOTIONREQ\_V1;
1166. typedef [switch\_type(DWORD)] union {
1167. [case(1)] DRS\_MSG\_FINISH\_DEMOTIONREQ\_V1 V1;
1168. } DRS\_MSG\_FINISH\_DEMOTIONREQ;
1169. typedef struct {
1170. DWORD dwOperationsDone;
1171. DWORD dwOpFailed;
1172. DWORD dwOpError;
1173. } DRS\_MSG\_FINISH\_DEMOTIONREPLY\_V1;
1174. typedef [switch\_type(DWORD)] union {
1175. [case(1)] DRS\_MSG\_FINISH\_DEMOTIONREPLY\_V1 V1;
1176. } DRS\_MSG\_FINISH\_DEMOTIONREPLY;
1177. typedef struct {
1178. [string] const WCHAR \*pwszCloneDCName;
1179. [string] const WCHAR \*pwszSite;
1180. } DRS\_MSG\_ADDCLONEDCREQ\_V1;
1181. typedef [switch\_type(DWORD)] union {
1182. [case(1)] DRS\_MSG\_ADDCLONEDCREQ\_V1 V1;
1183. } DRS\_MSG\_ADDCLONEDCREQ;
1184. typedef struct {
1185. [string] WCHAR \*pwszCloneDCName;
1186. [string] WCHAR \*pwszSite;
1187. [range(0,1024)] DWORD cPasswordLength;
1188. [size\_is(cPasswordLength)] WCHAR \*pwsNewDCAccountPassword;
1189. } DRS\_MSG\_ADDCLONEDCREPLY\_V1;
1190. typedef [switch\_type(DWORD)] union {
1191. [case(1)] DRS\_MSG\_ADDCLONEDCREPLY\_V1 V1;
1192. } DRS\_MSG\_ADDCLONEDCREPLY;
1193. typedef struct \_DRS\_MSG\_WRITENGCKEYREQ\_V1{
1194. [string] const WCHAR\* pwszAccount;
1195. [range(0,0xFFFF)] DWORD cNgcKey;
1196. [size\_is(cNgcKey)] UCHAR \* pNgcKey;
1197. } DRS\_MSG\_WRITENGCKEYREQ\_V1;
1198. typedef
1199. [switch\_type(DWORD)]
1200. union {
1201. [case(1)]
1202. DRS\_MSG\_WRITENGCKEYREQ\_V1 V1;
1203. } DRS\_MSG\_WRITENGCKEYREQ;
1204. typedef struct \_DRS\_MSG\_WRITENGCKEYREPLY\_V1{
1205. DWORD retVal;
1206. } DRS\_MSG\_WRITENGCKEYREPLY\_V1;
1207. typedef
1208. [switch\_type(DWORD)]
1209. union {
1210. [case(1)]
1211. DRS\_MSG\_WRITENGCKEYREPLY\_V1 V1;
1212. } DRS\_MSG\_WRITENGCKEYREPLY;
1213. typedef struct \_DRS\_MSG\_READNGCKEYREQ\_V1{
1214. [string] const WCHAR\* pwszAccount;
1215. } DRS\_MSG\_READNGCKEYREQ\_V1;
1216. typedef
1217. [switch\_type(DWORD)]
1218. union {
1219. [case(1)]
1220. DRS\_MSG\_READNGCKEYREQ\_V1 V1;
1221. } DRS\_MSG\_READNGCKEYREQ;
1222. typedef struct \_DRS\_MSG\_READNGCKEYREPLY\_V1{
1223. DWORD retVal;
1224. [range(0,0xFFFF)] DWORD cNgcKey;
1225. [size\_is(cNgcKey)] UCHAR \* pNgcKey;
1226. } DRS\_MSG\_READNGCKEYREPLY\_V1;
1227. typedef
1228. [switch\_type(DWORD)]
1229. union {
1230. [case(1)]
1231. DRS\_MSG\_READNGCKEYREPLY\_V1 V1;
1232. } DRS\_MSG\_READNGCKEYREPLY;

1. // opnum 0
2. ULONG
3. IDL\_DRSBind(
4. [in] handle\_t rpc\_handle,
5. [in, unique] UUID \*puuidClientDsa,
6. [in, unique] DRS\_EXTENSIONS \*pextClient,
7. [out] DRS\_EXTENSIONS \*\*ppextServer,
8. [out, ref] DRS\_HANDLE \*phDrs);
9. // opnum 1
10. ULONG
11. IDL\_DRSUnbind(
12. [in, out, ref] DRS\_HANDLE \*phDrs);
13. // opnum 2
14. ULONG
15. IDL\_DRSReplicaSync(
16. [in, ref] DRS\_HANDLE hDrs,
17. [in] DWORD dwVersion,
18. [in, ref, switch\_is(dwVersion)] DRS\_MSG\_REPSYNC \*pmsgSync);
19. // opnum 3
20. ULONG
21. IDL\_DRSGetNCChanges(
22. [in, ref] DRS\_HANDLE hDrs,
23. [in] DWORD dwInVersion,
24. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_GETCHGREQ \*pmsgIn,
25. [out, ref] DWORD \*pdwOutVersion,
26. [out, ref, switch\_is(\*pdwOutVersion)]
27. DRS\_MSG\_GETCHGREPLY \*pmsgOut);
28. // opnum 4
29. ULONG
30. IDL\_DRSUpdateRefs(
31. [in, ref] DRS\_HANDLE hDrs,
32. [in] DWORD dwVersion,
33. [in, ref, switch\_is(dwVersion)] DRS\_MSG\_UPDREFS \*pmsgUpdRefs);
34. // opnum 5
35. ULONG
36. IDL\_DRSReplicaAdd(
37. [in, ref] DRS\_HANDLE hDrs,
38. [in] DWORD dwVersion,
39. [in, ref, switch\_is(dwVersion)] DRS\_MSG\_REPADD \*pmsgAdd);
40. // opnum 6
41. ULONG
42. IDL\_DRSReplicaDel(
43. [in, ref] DRS\_HANDLE hDrs,
44. [in] DWORD dwVersion,
45. [in, ref, switch\_is(dwVersion)] DRS\_MSG\_REPDEL \*pmsgDel);
46. // opnum 7
47. ULONG
48. IDL\_DRSReplicaModify(
49. [in, ref] DRS\_HANDLE hDrs,
50. [in] DWORD dwVersion,
51. [in, ref, switch\_is(dwVersion)] DRS\_MSG\_REPMOD \*pmsgMod);
52. // opnum 8
53. ULONG
54. IDL\_DRSVerifyNames(
55. [in, ref] DRS\_HANDLE hDrs,
56. [in] DWORD dwInVersion,
57. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_VERIFYREQ \*pmsgIn,
58. [out, ref] DWORD \*pdwOutVersion,
59. [out, ref, switch\_is(\*pdwOutVersion)]
60. DRS\_MSG\_VERIFYREPLY \*pmsgOut);
61. // opnum 9
62. ULONG
63. IDL\_DRSGetMemberships(
64. [in, ref] DRS\_HANDLE hDrs,
65. [in] DWORD dwInVersion,
66. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_REVMEMB\_REQ \*pmsgIn,
67. [out, ref] DWORD \*pdwOutVersion,
68. [out, ref, switch\_is(\*pdwOutVersion)]
69. DRS\_MSG\_REVMEMB\_REPLY \*pmsgOut);
70. // opnum 10
71. ULONG
72. IDL\_DRSInterDomainMove(
73. [in, ref] DRS\_HANDLE hDrs,
74. [in] DWORD dwInVersion,
75. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_MOVEREQ \*pmsgIn,
76. [out, ref] DWORD \*pdwOutVersion,
77. [out, ref, switch\_is(\*pdwOutVersion)] DRS\_MSG\_MOVEREPLY \*pmsgOut);
78. // opnum 11
79. ULONG
80. IDL\_DRSGetNT4ChangeLog(
81. [in, ref] DRS\_HANDLE hDrs,
82. [in] DWORD dwInVersion,
83. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_NT4\_CHGLOG\_REQ \*pmsgIn,
84. [out, ref] DWORD \*pdwOutVersion,
85. [out, ref, switch\_is(\*pdwOutVersion)]
86. DRS\_MSG\_NT4\_CHGLOG\_REPLY \*pmsgOut);
87. // opnum 12
88. ULONG
89. IDL\_DRSCrackNames(
90. [in, ref] DRS\_HANDLE hDrs,
91. [in] DWORD dwInVersion,
92. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_CRACKREQ \*pmsgIn,
93. [out, ref] DWORD \*pdwOutVersion,
94. [out, ref, switch\_is(\*pdwOutVersion)]
95. DRS\_MSG\_CRACKREPLY \*pmsgOut);
96. // opnum 13
97. ULONG
98. IDL\_DRSWriteSPN(
99. [in, ref] DRS\_HANDLE hDrs,
100. [in] DWORD dwInVersion,
101. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_SPNREQ \*pmsgIn,
102. [out, ref] DWORD \*pdwOutVersion,
103. [out, ref, switch\_is(\*pdwOutVersion)] DRS\_MSG\_SPNREPLY \*pmsgOut);
104. // opnum 14
105. ULONG
106. IDL\_DRSRemoveDsServer(
107. [in, ref] DRS\_HANDLE hDrs,
108. [in] DWORD dwInVersion,
109. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_RMSVRREQ \*pmsgIn,
110. [out, ref] DWORD \*pdwOutVersion,
111. [out, ref, switch\_is(\*pdwOutVersion)]
112. DRS\_MSG\_RMSVRREPLY \*pmsgOut);
113. // opnum 15
114. ULONG
115. IDL\_DRSRemoveDsDomain(
116. [in, ref] DRS\_HANDLE hDrs,
117. [in] DWORD dwInVersion,
118. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_RMDMNREQ \*pmsgIn,
119. [out, ref] DWORD \*pdwOutVersion,
120. [out, ref, switch\_is(\*pdwOutVersion)]
121. DRS\_MSG\_RMDMNREPLY \*pmsgOut);
122. // opnum 16
123. ULONG
124. IDL\_DRSDomainControllerInfo(
125. [in, ref] DRS\_HANDLE hDrs,
126. [in] DWORD dwInVersion,
127. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_DCINFOREQ \*pmsgIn,
128. [out, ref] DWORD \*pdwOutVersion,
129. [out, ref, switch\_is(\*pdwOutVersion)]
130. DRS\_MSG\_DCINFOREPLY \*pmsgOut);
131. // opnum 17
132. ULONG
133. IDL\_DRSAddEntry(
134. [in, ref] DRS\_HANDLE hDrs,
135. [in] DWORD dwInVersion,
136. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_ADDENTRYREQ \*pmsgIn,
137. [out, ref] DWORD \*pdwOutVersion,
138. [out, ref, switch\_is(\*pdwOutVersion)]
139. DRS\_MSG\_ADDENTRYREPLY \*pmsgOut);
140. // opnum 18
141. ULONG
142. IDL\_DRSExecuteKCC(
143. [in, ref] DRS\_HANDLE hDrs,
144. [in] DWORD dwInVersion,
145. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_KCC\_EXECUTE \*pmsgIn);
146. // opnum 19
147. ULONG
148. IDL\_DRSGetReplInfo(
149. [in, ref] DRS\_HANDLE hDrs,
150. [in] DWORD dwInVersion,
151. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_GETREPLINFO\_REQ \*pmsgIn,
152. [out, ref] DWORD \*pdwOutVersion,
153. [out, ref, switch\_is(\*pdwOutVersion)]
154. DRS\_MSG\_GETREPLINFO\_REPLY \*pmsgOut);
155. // opnum 20
156. ULONG
157. IDL\_DRSAddSidHistory(
158. [in, ref] DRS\_HANDLE hDrs,
159. [in] DWORD dwInVersion,
160. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_ADDSIDREQ \*pmsgIn,
161. [out, ref] DWORD \*pdwOutVersion,
162. [out, ref, switch\_is(\*pdwOutVersion)]
163. DRS\_MSG\_ADDSIDREPLY \*pmsgOut);
164. // opnum 21
165. ULONG
166. IDL\_DRSGetMemberships2(
167. [in, ref] DRS\_HANDLE hDrs,
168. [in] DWORD dwInVersion,
169. [in, ref, switch\_is(dwInVersion)]
170. DRS\_MSG\_GETMEMBERSHIPS2\_REQ \*pmsgIn,
171. [out, ref] DWORD \*pdwOutVersion,
172. [out, ref, switch\_is(\*pdwOutVersion)]
173. DRS\_MSG\_GETMEMBERSHIPS2\_REPLY \*pmsgOut);
174. // opnum 22
175. ULONG
176. IDL\_DRSReplicaVerifyObjects(
177. [in, ref] DRS\_HANDLE hDrs,
178. [in] DWORD dwVersion,
179. [in, ref, switch\_is(dwVersion)] DRS\_MSG\_REPVERIFYOBJ \*pmsgVerify);
180. // opnum 23
181. ULONG
182. IDL\_DRSGetObjectExistence (
183. [in, ref] DRS\_HANDLE hDrs,
184. [in] DWORD dwInVersion,
185. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_EXISTREQ \*pmsgIn,
186. [out, ref] DWORD \*pdwOutVersion,
187. [out, ref, switch\_is(\*pdwOutVersion)]
188. DRS\_MSG\_EXISTREPLY \*pmsgOut);
189. // opnum 24
190. ULONG
191. IDL\_DRSQuerySitesByCost(
192. [in, ref] DRS\_HANDLE hDrs,
193. [in] DWORD dwInVersion,
194. [in, ref, switch\_is(dwInVersion)] DRS\_MSG\_QUERYSITESREQ \*pmsgIn,
195. [out, ref] DWORD \*pdwOutVersion,
196. [out, ref, switch\_is(\*pdwOutVersion)]
197. DRS\_MSG\_QUERYSITESREPLY \*pmsgOut);
198. // opnum 25
199. ULONG
200. IDL\_DRSInitDemotion(
201. [in, ref] DRS\_HANDLE hDrs,
202. [in] DWORD dwInVersion,
203. [in, ref, switch\_is(dwInVersion)]
204. DRS\_MSG\_INIT\_DEMOTIONREQ\* pmsgIn,
205. [out, ref] DWORD\* pdwOutVersion,
206. [out, ref, switch\_is(\*pdwOutVersion)]
207. DRS\_MSG\_INIT\_DEMOTIONREPLY\* pmsgOut);
208. // opnum 26
209. ULONG
210. IDL\_DRSReplicaDemotion(
211. [in, ref] DRS\_HANDLE hDrs,
212. [in] DWORD dwInVersion,
213. [in, ref, switch\_is(dwInVersion)]
214. DRS\_MSG\_REPLICA\_DEMOTIONREQ\* pmsgIn,
215. [out, ref] DWORD\* pdwOutVersion,
216. [out, ref, switch\_is(\*pdwOutVersion)]
217. DRS\_MSG\_REPLICA\_DEMOTIONREPLY\* pmsgOut);
218. // opnum 27
219. ULONG
220. IDL\_DRSFinishDemotion(
221. [in, ref] DRS\_HANDLE hDrs,
222. [in] DWORD dwInVersion,
223. [in, ref, switch\_is(dwInVersion)]
224. DRS\_MSG\_FINISH\_DEMOTIONREQ\* pmsgIn,
225. [out, ref] DWORD\* pdwOutVersion,
226. [out, ref, switch\_is(\*pdwOutVersion)]
227. DRS\_MSG\_FINISH\_DEMOTIONREPLY\* pmsgOut);
228. // opnum 28
229. ULONG
230. IDL\_DRSAddCloneDC (
231. [in, ref] DRS\_HANDLE hDrs,
232. [in] DWORD dwInVersion,
233. [in, ref, switch\_is(dwInVersion)]
234. DRS\_MSG\_ADDCLONEDCREQ\* pmsgIn,
235. [out, ref] DWORD \* pdwOutVersion,
236. [out, ref, switch\_is(\*pdwOutVersion)]
237. DRS\_MSG\_ADDCLONEDCREPLY\* pmsgOut
238. );
239. // opnum 29
240. ULONG
241. IDL\_DRSWriteNgcKey(
242. [in, ref] DRS\_HANDLE hDrs,
243. [in] DWORD dwInVersion,
244. [in, ref, switch\_is(dwInVersion)]
245. DRS\_MSG\_WRITENGCKEYREQ\* pmsgIn,
246. [out, ref] DWORD\* pdwOutVersion,
247. [out, ref, switch\_is(\*pdwOutVersion)]
248. DRS\_MSG\_WRITENGCKEYREPLY\* pmsgOut
249. );
250. // opnum 30
251. ULONG
252. IDL\_DRSReadNgcKey(
253. [in, ref] DRS\_HANDLE hDrs,
254. [in] DWORD dwInVersion,
255. [in, ref, switch\_is(dwInVersion)]
256. DRS\_MSG\_READNGCKEYREQ\* pmsgIn,
257. [out, ref] DWORD\* pdwOutVersion,
258. [out, ref, switch\_is(\*pdwOutVersion)]
259. DRS\_MSG\_READNGCKEYREPLY\* pmsgOut
260. );
261. }
262. // This is the "real" ntdscript interface.
263. [
264. uuid(7c44d7d4-31d5-424c-bd5e-2b3e1f323d22), version(1.0),
265. pointer\_default (unique)
266. ]
267. interface dsaop
268. {
269. typedef struct {
270. DWORD Flags;
271. [range(1,1024)] DWORD cbPassword;
272. [size\_is(cbPassword)] BYTE \*pbPassword;
273. } DSA\_MSG\_EXECUTE\_SCRIPT\_REQ\_V1;
274. typedef [switch\_type(DWORD)] union {
275. [case(1)] DSA\_MSG\_EXECUTE\_SCRIPT\_REQ\_V1 V1;
276. } DSA\_MSG\_EXECUTE\_SCRIPT\_REQ;
277. typedef struct {
278. DWORD dwOperationStatus;
279. [string] LPWSTR pwErrMessage;
280. } DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY\_V1;
281. typedef [switch\_type(DWORD)] union {
282. [case(1)] DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY\_V1 V1;
283. } DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY;
284. typedef struct {
285. DWORD Reserved;
286. } DSA\_MSG\_PREPARE\_SCRIPT\_REQ\_V1;
287. typedef [switch\_type(DWORD)] union {
288. [case(1)] DSA\_MSG\_PREPARE\_SCRIPT\_REQ\_V1 V1;
289. } DSA\_MSG\_PREPARE\_SCRIPT\_REQ;
290. typedef struct {
291. DWORD dwOperationStatus;
292. [string] LPWSTR pwErrMessage;
293. [range(0,1024)] DWORD cbPassword;
294. [size\_is(cbPassword)] BYTE \*pbPassword;
295. [range(0,10485760)] DWORD cbHashBody;
296. [size\_is(cbHashBody)] BYTE \*pbHashBody;
297. [range(0,10485760)] DWORD cbHashSignature;
298. [size\_is(cbHashSignature)] BYTE \*pbHashSignature;
299. } DSA\_MSG\_PREPARE\_SCRIPT\_REPLY\_V1;
300. typedef [switch\_type(DWORD)] union {
301. [case(1)] DSA\_MSG\_PREPARE\_SCRIPT\_REPLY\_V1 V1;
302. } DSA\_MSG\_PREPARE\_SCRIPT\_REPLY;
303. // opnum 0
304. ULONG
305. IDL\_DSAPrepareScript(
306. [in] handle\_t hRpc,
307. [in] DWORD dwInVersion,
308. [in, ref, switch\_is(dwInVersion)]
309. DSA\_MSG\_PREPARE\_SCRIPT\_REQ \*pmsgIn,
310. [out, ref] DWORD \*pdwOutVersion,
311. [out, ref, switch\_is(\*pdwOutVersion)]
312. DSA\_MSG\_PREPARE\_SCRIPT\_REPLY \*pmsgOut);
313. // opnum 1
314. ULONG
315. IDL\_DSAExecuteScript(
316. [in] handle\_t hRpc,
317. [in] DWORD dwInVersion,
318. [in, ref, switch\_is(dwInVersion)]
319. DSA\_MSG\_EXECUTE\_SCRIPT\_REQ \*pmsgIn,
320. [out, ref] DWORD \*pdwOutVersion,
321. [out, ref, switch\_is(\*pdwOutVersion)]
322. DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY \*pmsgOut);
323. }

# 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.

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.

The following tables show the relationships between Microsoft product versions or supplemental software and the roles they perform.

| Windows Client releases | Client role | Server role |
| --- | --- | --- |
| Windows 2000 Professional operating system | Yes | Yes |
| Windows XP operating system | Yes | Yes |
| Windows Vista operating system | Yes | Yes |
| Windows 7 operating system | Yes | Yes |
| Windows 8 operating system | Yes | Yes |
| Windows 8.1 operating system | Yes | Yes |
| Windows 10 operating system | Yes | Yes |

| Windows Server releases | Client role | Server role |
| --- | --- | --- |
| Windows 2000 Server operating system | Yes | Yes |
| Windows Server 2003 operating system | Yes | Yes |
| Windows Server 2003 R2 operating system | Yes | Yes |
| Windows Server 2008 operating system | Yes | Yes |
| Windows Server 2008 R2 operating system | Yes | Yes |
| Windows Server 2012 operating system | Yes | Yes |
| Windows Server 2012 R2 operating system | Yes | Yes |
| Windows Server 2016 operating system | Yes | Yes |
| Windows Server operating system | Yes | Yes |

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 2.1](#Appendix_A_Target_1): Applicable Windows Server releases listen only on the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331)-over-TCP protocol sequence. Windows clients attempt to connect using only the RPC-over-TCP protocol sequence.

[<2> Section 2.2.2](#Appendix_A_Target_2): Windows 2000 Server, Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, and Windows Server 2008 R2 [**AD DS**](#gt_2e72eeeb-aee9-4b0a-adc6-4476bacf5024) [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) do not store the "RPC" [**SPN**](#gt_547217ca-134f-4b43-b375-f5bca4c16ce4).

[<3> Section 2.2.3](#Appendix_A_Target_3): Windows implements DC-to-DC interaction with an SPN with [**service class**](#gt_c1386afd-9d42-47c7-a7ea-d01912a17647) "E3514235-4B06-11D1-AB04-00C04FC2DCD2". See [DRS\_SPN\_CLASS](#Section_3e0e2ef55dc54b37a927323f3c4876ca).

[<4> Section 2.2.4.2](#Appendix_A_Target_4): SPN "ldap/<NetBIOS hostname>/<NetBIOS domain name>" is available in Windows Server 2008 R2 and later.

[<5> Section 4.1](#Appendix_A_Target_5): All [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) methods and their associated [**concrete types**](#gt_cd539538-9f7e-4881-b5af-2301b420244d) have existed in the drsuapi RPC interface since Windows 2000 operating system except those listed in the following table. All IDL methods and their associated concrete types continue to exist in this interface in subsequent versions of Windows according to the applicability list at the beginning of this section.

| Data type or IDL method | Section | Introduced in Windows Server release |
| --- | --- | --- |
| DRS\_MSG\_GETCHGREQ\_V7 support | [4.1.10.2.5](#Section_5ef4f597a3974f6fa98b7a034247d886) | Windows Server 2003 |
| DRS\_MSG\_GETREPLINFO\_REQ\_V2 | [4.1.13.1.3](#Section_a5f01efdecd942c6a0c78cf67d028589) | Windows Server 2003 |
| DRS\_MSG\_DCINFOREPLY\_V3 | [4.1.5.1.6](#Section_cafc7232c6da478484d7e5d8c804c2d9) | Windows Server 2008 |
| DS\_DOMAIN\_CONTROLLER\_INFO\_3W | [4.1.5.1.10](#Section_08f99ee78235482bbfe5c6170f133cd4) | Windows Server 2008 |
| DRS\_MSG\_GETCHGREQ\_V8 support | [4.1.10.2.6](#Section_4304bb4ae9b54c8a8731df4d6f9ab567) | Windows Server 2003 |
| DRS\_MSG\_GETCHGREQ\_V10 support | [4.1.10.2.7](#Section_92b1b77d205846e09e8c6664b96a0cf9) | Windows Server 2008 R2 |
| DRS\_MSG\_GETCHGREPLY\_V6 support | [4.1.10.2.12](#Section_1317a6545dd645ffaf73919cbc7fbb45) | Windows Server 2003 |
| DRS\_MSG\_GETCHGREPLY\_V7 support | [4.1.10.2.13](#Section_26eaca610f1947e7b3042580e9870aa8) | Windows Server 2003 |
| DRS\_MSG\_GETCHGREPLY\_V9 support | [4.1.10.2.14](#Section_b9564a194500444ba99b0da1b08cdb6f) | Windows Server 2016 |
| DRS\_MSG\_ADDENTRYREQ\_V3 support | [4.1.1.1.4](#Section_d04ae3cc4c314c9ebb245f718a6ee646) | Windows Server 2003 |
| DRS\_MSG\_ADDENTRYREPLY\_V3 support | [4.1.1.1.8](#Section_1eeb493e93f1424cb8ebca74e6f051a0) | Windows Server 2003 |
| IDL\_DRSGetObjectExistence | [4.1.12](#Section_6355d4f5f5564527adde37afba2fcf56) | Windows Server 2003 |
| IDL\_DRSReplicaVerifyObjects | [4.1.24](#Section_8dba150d50f647f1941e1a606c30db0b) | Windows Server 2003 |
| IDL\_DRSFinishDemotion | [4.1.7](#Section_0bf530e81be04f48b8c2208031a8725f) | Windows Server 2008 |
| IDL\_DRSInitDemotion | [4.1.14](#Section_faca17da3f7f409298dbfd2ce7d98b8c) | Windows Server 2008 |
| IDL\_DRSReplicaDemotion | [4.1.21](#Section_8a2f0388bdfb4519a8c3384f27c11639) | Windows Server 2008 |
| IDL\_DRSAddCloneDC | [4.1.29](#Section_ef0bfb1d037b4626a6d9cc7589bc5786) | Windows Server 2012 |
| IDL\_DRSWriteNgcKey | [4.1.30](#Section_7a140389caa34718bb1ad64483933eb0) | Windows Server 2016 |
| IDL\_DRSReadNgcKey | [4.1.31](#Section_a80c60ac9864444a95136c0c894fbb8d) | Windows Server 2016 |
| DRS\_MSG\_GETCHGREQ\_V11 support | [4.1.10.2.8](#Section_cb2bab15950b48f8af00118e186a1311) | Windows Server v1803 operating system |
| DRS\_MSG\_REPADD\_V3 support | [4.1.19.1.4](#Section_63f7638bc9514486ab3f3f086d9fb622) | Windows Server v1803 |
| DRS\_MSG\_REPSYNC\_V2 support | [4.1.23.1.3](#Section_f32ab3844894416793f0eda08703d76b) | Windows Server v1803 |
| DRS\_MSG\_UPDREFS\_V2 support | [4.1.26.1.3](#Section_af174769feae42128e1ca6072b022e6c) | Windows Server v1803 |

[<6> Section 4.1.1.1.2](#Appendix_A_Target_6): Though this request version appears in the IDL, Windows DCs do not support it. It was never supported in any of the applicable Windows Server releases.

[<7> Section 4.1.1.1.6](#Appendix_A_Target_7): Though this response version appears in the IDL, Windows DCs do not support it.

[<8> Section 4.1.1.3](#Appendix_A_Target_8): This operation is only supported by [**AD LDS**](#gt_afdbd6cd-9f55-4d2f-a98e-1207985534ab) and AD DS in Windows Server 2008 and later.

[<9> Section 4.1.2.2.6](#Appendix_A_Target_9): The function determines whether auditing is enabled on the server by querying the LSA information policy on the server associated with *ctx* and by confirming that the information policy is set to generate both success and failure audits for the "account management" audit category. To achieve this, the LsarOpenPolicy2, LsarQueryInformationPolicy, and LsarClose messages in [[MS-LSAD]](%5bMS-LSAD%5d.pdf#Section_1b5471ef4c334a91b079dfcbb82f05cc) are used ([MS-LSAD] sections 3.1.4.4.1, 3.1.4.4.4, and 3.1.4.9.4). The **srcDomainController** variable in the [IDL\_DRSAddSidHistory](#Section_376230a5d8064ae5970af6243ee193c8) method is used as the *SystemName* parameter to LsarOpenPolicy2, and the *DesiredAccess* parameter to LsarOpenPolicy2 is set to (POLICY\_VIEW\_AUDIT\_INFORMATION + POLICY\_VIEW\_LOCAL\_INFORMATION). On success, the *PolicyHandle* acquired from the LsarOpenPolicy2 message is passed to LsarQueryInformationPolicy with PolicyAuditEventsInformation as the information [**class**](#gt_18393bbe-0c06-42b7-890d-b94a9a40b6e0). The check to determine whether success and failure audits are enabled for "account management" is achieved by performing the following evaluation:

1. PolicyInformation^.PolicyAuditEventsInfo.EventAuditingOptions[6] ∩
2. { POLICY\_AUDIT\_EVENT\_SUCCESS, POLICY\_AUDIT\_EVENT\_FAILURE } =
3. { POLICY\_AUDIT\_EVENT\_SUCCESS, POLICY\_AUDIT\_EVENT\_FAILURE }

where *PolicyInformation* is the result from the LsarQueryInformationPolicy message. *PolicyHandle* is then closed by using the LsarClose message.

The function generates an audit on the DC associated with *ctx* by adding the source [**principal**](#gt_8492780e-99e2-47ba-8553-aedb8de9f9c0) *(pmsgIn^.V1.SrcPrincipal*, where *pmsgIn* is a parameter to the IDL\_DRSAddSidHistory method, which in turn calls this method) to the [**group**](#gt_51c51c14-7f9d-4c0b-a69c-d3e059bfffac) *srcDomainFlatName*$$$ on the DC associated with *ctx*, where *srcDomainFlatName* is the NetBIOS name of the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) to which the source principal belongs. After adding the principal to the group, it then removes the principal from the group, leaving the group in its original state but having generated an audit event as a side effect of manipulating the group's membership.

[<10> Section 4.1.2.2.13](#Appendix_A_Target_10): This test is implemented in two steps. First, it is determined if the DC associated with *ctx* is running at least Windows 2000. This is determined by whether a SamrConnect5 or SamrConnect4 API call (as specified in [[MS-SAMR]](%5bMS-SAMR%5d.pdf#Section_4df07fab1bbc452f8e927853a3c7e380)) to the DC is successful. If it is, the DC is running at least Windows 2000, and the function returns true.

Otherwise, the DC is considered to be running Windows NT 4.0 operating system. The function then connects to the registry service on the DC named in *ctx* and queries the value of the "CSDVersion" registry value on the "HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion" registry key. If the value is not equal to any of the following strings, the function returns true; otherwise, it returns false:

* Service Pack 0
* Service Pack 1
* Service Pack 2
* Service Pack 3

If the registry service on the DC could not be contacted, or if the registry key or registry value does not exist, the function returns false.

[<11> Section 4.1.3.1](#Appendix_A_Target_11): Windows non-DC client callers always pass NTDSAPI\_CLIENT\_GUID in *puuidClientDsa*. If a Windows DC client caller uses the returned DRS\_HANDLE for subsequent calls to the [IDL\_DRSWriteSPN](#Section_8b129dc8ed4545379555b6fef764ab7d) method, then the client passes NTDSAPI\_CLIENT\_GUID in *puuidClientDsa*. In any other cases, Windows DC client callers pass DC!serverGuid in *puuidClientDsa*.

[<12> Section 4.1.3.1](#Appendix_A_Target_12): Windows non-DC client callers always set the **dwFlags** field of the [DRS\_EXTENSIONS\_INT](#Section_3ee529b123db4996948a042f04998e91) structure to zero. Windows non-DC client callers always set the **SiteObjGuid** field of the DRS\_EXTENSIONS\_INT structure to the [**NULL GUID**](#gt_ba500a5b-8c29-467c-a335-0980c8b11304) value. Windows non-DC client callers always set the **Pid** field of the DRS\_EXTENSIONS\_INT structure to an implementation-specific, client-local process identifier (PID).

Windows non-DC clients do not include the following fields in the DRS\_EXTENSIONS\_INT structure. In addition, the following fields are included by Windows DC clients only if the clients are running the corresponding versions of Windows.

| Field | Included by |
| --- | --- |
| **ConfigObjGUID** | Windows DC clients running Windows Server 2003 R2 (AD LDS only) and Windows Server 2008 and later. |
| **dwFlagsExt** | Windows DC clients running Windows Server 2003 R2 (AD LDS only) and Windows Server 2008 and later. |
| **dwExtCaps** | Windows DC clients running Windows Server 2012 R2 and later. |

[<13> Section 4.1.3.2](#Appendix_A_Target_13): The **ConfigObjGUID** and **dwFlagsExt** fields in the DRS\_EXTENSIONS\_INT structure are included only by servers running Windows Server 2003 R2 (AD LDS only) and Windows Server 2008 and later.

[<14> Section 4.1.5.2](#Appendix_A_Target_14): All of the information levels listed in section [4.1.5.2](#Section_120171ab5a8a42a3a75251749d6a0130) have existed in the [drsuapi RPC interface](#Section_58f33216d9f143bfa18387e3c899c410) since Windows 2000, except as noted in the following table.

| Infolevel | Introduced in Windows Server release |
| --- | --- |
| 3 | Windows Server 2008 |

[<15> Section 4.1.6.3](#Appendix_A_Target_15): The Windows implementation of this method puts the [**KCC**](#gt_c7d4f1f6-5285-4168-b21a-022f775a3f58) execution requests in a local-machine work queue. If DS\_KCC\_FLAG\_DAMPED is specified in the call to [IDL\_DRSExecuteKCC](#Section_ad807917687b40d9abe2053af0246523) and there is already a request pending, the execution request is not added to the queue in order to reduce redundant requests.

[<16> Section 4.1.7.3](#Appendix_A_Target_16): The Windows implementation of the IDL\_DRSFinishDemotion method causes the underlying RPC protocol [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) to throw an RPC\_S\_INVALID\_TAG exception when returning ERROR\_INVALID\_PARAMETER.

[<17> Section 4.1.10.1.1](#Appendix_A_Target_17): The Windows implementation never declares msgIn.uuidInvocIdSrc and msgIn.usnvecFrom that are otherwise valid to be stale.

[<18> Section 4.1.10.1.1](#Appendix_A_Target_18): The internal format of [USN\_VECTOR](#Section_595d11b86ca74a61bd563e6a2b99b76b) identifies the start of a [**cycle**](#gt_e14454ba-5d3b-4fdb-99e5-50ecf632bd16).

[<19> Section 4.1.10.1.2](#Appendix_A_Target_19): The goal is advanced on each request of a cycle.

[<20> Section 4.1.10.1.2](#Appendix_A_Target_20): *c.usnHighPropUpdate* is never set to 0.

[<21> Section 4.1.10.1.3](#Appendix_A_Target_21): The Request Role [**extended operation**](#gt_8514d343-000a-45e2-918b-5f6100e2e7c6) is supported by Windows 2000 Server and later.

[<22> Section 4.1.10.1.3](#Appendix_A_Target_22): The Abandon Role extended operation is supported by Windows 2000 Server and later.

[<23> Section 4.1.10.1.3](#Appendix_A_Target_23): The Allocate [**RIDs**](#gt_df3d0b61-56cd-4dac-9402-982f1fedc41c) extended operation is supported by Windows 2000 Server and later.

[<24> Section 4.1.10.1.3](#Appendix_A_Target_24): The Replicate Single Object extended operation is supported by Windows Server 2003 and later.

[<25> Section 4.1.10.1.3](#Appendix_A_Target_25): The Replicate Single Object including Secret Data extended operation is supported by Windows Server 2008 and later.

[<26> Section 4.1.10.2.2](#Appendix_A_Target_26): Though this request version appears in the IDL, Windows DCs never send this request version by means of RPC. It exists solely to support SMTP [**replication**](#gt_a5678f3c-cf60-4b89-b835-16d643d1debb) (see [[MS-SRPL]](%5bMS-SRPL%5d.pdf#Section_ec69eea50d5e428ab5bc66732aaeb866)).

[<27> Section 4.1.10.2.3](#Appendix_A_Target_27): Although this request version appears in the IDL, Windows DCs never send this request version using RPC. It exists solely to support SMTP replication ([MS-SRPL]).

[<28> Section 4.1.10.2.5](#Appendix_A_Target_28): Though this request version appears in the IDL, Windows DCs never send this request version using RPC. It exists solely to support SMTP replication ([MS-SRPL]).

[<29> Section 4.1.10.2.19](#Appendix_A_Target_29): Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, and Windows Server 2008 R2 enforce the following range for the **cbCompressedSize** member: "[range(1,10485760)]".

[<30> Section 4.1.10.5.7](#Appendix_A_Target_30): The server tests the response against the limits after adding each [**object**](#gt_8bb43a65-7a8c-4585-a7ed-23044772f8ca) and link to the response, unless the object is a [**parent object**](#gt_0d41951a-62f0-4fbd-bb23-22f645ae3bf5) that is being included because of the Ancestors predicate (see the [GetReplChanges](#Section_b13f8ab038b447dbbb55e188f776673b) method). If the test shows that the response has exceeded one of the limits, the server stops adding to the response. The server might return more objects or bytes than the limits.

[<31> Section 4.1.10.6.6](#Appendix_A_Target_31): Windows [DCs](#Section_c7f1df49ea514e1aa8af063ac3f5e219) assign the remainder of the bits in values of *o*!instanceType for a given object *o* as follows:

* IT\_UNINSTANT: Set if and only if *o* is an [**NC**](#gt_784c7cce-f782-48d8-9444-c9030ba86942) root and its [**NC replica**](#gt_325d116f-cdbe-4dbd-b7e6-769ba75bf210) is not present on the DC.
* IT\_NC\_ABOVE: Set if and only if *o* is an NC root and the DC has an NC replica with NC root *p* such that *p* is the parent of *o*.
* IT\_NC\_COMING: Set if and only if *o* is an NC root and the DC has not yet completed the first replication cycle for that NC replica.
* IT\_NC\_GOING: Set if and only if *o* is an NC root and the DC is in the process of removing its [**replica**](#gt_ea02e669-2dda-460c-9992-b12a23caeeac) of the NC.

[<32> Section 4.1.12.3](#Appendix_A_Target_32): Windows uses count = 1000.

[<33> Section 4.1.14.2](#Appendix_A_Target_33): The Windows implementation of the **IDL\_DRSInitDemotion** method causes the underlying RPC protocol (as specified in [MS-RPCE]) to throw an RPC\_S\_INVALID\_TAG exception when returning ERROR\_ACCESS\_DENIED.

[<34> Section 4.1.15.1.2](#Appendix_A_Target_34): Although this request version appears in the IDL, Windows DCs do not support it. It was never supported in any of the applicable Windows Server releases.

[<35> Section 4.1.15.1.5](#Appendix_A_Target_35): Although this response version appears in the IDL, Windows DCs do not support it.

[<36> Section 4.1.16.3](#Appendix_A_Target_36): The server returns the error ERROR\_DS\_GENERIC\_ERROR if the Intersite Messaging Service is not running on the server.

[<37> Section 4.1.24.3](#Appendix_A_Target_37): The Windows implementation of the for loop uses IDL\_DRSGetObjectExistence to determine if object *o* exists at *refDsa*, and logs a message to the Windows Event Log.

[<38> Section 5.39](#Appendix_A_Target_38): In Windows 2000 Server, the **cb** field contains the count of bytes in the fields **dwFlags** through **Pid**, inclusive, which is the size of the structure in that version minus the 4 bytes of the **cb** field.

[<39> Section 5.39](#Appendix_A_Target_39): In AD DS and AD LDS servers running Windows Server 2003, and in AD DS servers running Windows Server 2003 R2, the **cb** field contains the count of bytes in the fields **dwFlags** through **dwReplEpoch**, inclusive, which is the size of the structure in those versions minus the 4 bytes of the **cb** field.

[<40> Section 5.39](#Appendix_A_Target_40): In Windows Server 2003 R2 (AD LDS only), Windows Server 2008, Windows Server 2008 R2, and Windows Server 2012, the **cb** field contains the count of bytes in the fields **dwFlags** through **ConfigObjGUID**, inclusive, which is the size of the structure in those versions minus the 4 bytes of the **cb** field.

[<41> Section 5.39](#Appendix_A_Target_41): Client callers set dwFlags to zero.

[<42> Section 5.39](#Appendix_A_Target_42): This field contains the process ID of the client.

[<43> Section 5.39](#Appendix_A_Target_43): The **dwReplEpoch** field in the DRS\_EXTENSIONS\_INT structure is included only by servers running Windows Server 2003 and later.

[<44> Section 5.39](#Appendix_A_Target_44): The **ConfigObjGUID** and **dwFlagsExt** fields in the DRS\_EXTENSIONS\_INT structure are included only by AD DS servers running Windows Server 2008 and later, and by AD LDS servers running Windows Server 2003 R2 and later.

[<45> Section 5.39](#Appendix_A_Target_45): The **ConfigObjGUID** and **dwFlagsExt** fields in the DRS\_EXTENSIONS\_INT structure are included only by AD DS servers running Windows Server 2008 and later, and by AD LDS servers running Windows Server 2003 R2 and later.

[<46> Section 5.39](#Appendix_A_Target_46): The **dwExtCaps** field in the DRS\_EXTENSIONS\_INT structure is included only by AD DS and AD LDS servers running Windows Server 2012 R2 and later.

[<47> Section 5.41](#Appendix_A_Target_47): All the DRS\_OPTIONS listed in section [5.41](#Section_AC9C8A11CD464080ACBF9FAA86344030) have existed in the drsuapi RPC interface since Windows 2000 except those listed in the following table.

| Flag | Introduced in Windows Server release |
| --- | --- |
| DRS\_SYNC\_PAS | Windows Server 2003 |

This flag is ignored inbound on the operating systems that are prior to its introduction.

All the DRS\_OPTIONS listed in section 5.41 continue to exist in this interface in subsequent versions of Windows according to the applicability list at the beginning of this section, except those listed in the following table.

| Flag | Deprecated in Windows Server release |
| --- | --- |
| DRS\_SYNC\_ALL | Windows Server 2012 |

This flag is not to be used on the operating system from which it was removed or on subsequent versions of the operating system. If this flag is used on a deprecated platform, the error code ERROR\_DS\_DRA\_INVALID\_PARAMETER will be returned.

The following pseudocode specifies how to unmask the flags that are unsupported for each operating system.

1. msgReq.ulFlags &= msgReq.ulFlags & allowedFlagsForSpecificOS

[<48> Section 5.50](#Appendix_A_Target_48): No range is supported on any member of DSNAME in Windows 2000 Server. A range of 0 to 10485760 is supported on the **NameLen** member of DSNAME in Windows Server 2003 and Windows Server 2003 R2. A range of 0 to 10485761 is supported on the **StringName** member of DSNAME in Windows Server 2008 and later.

[<49> Section 5.170](#Appendix_A_Target_49): Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 AD DS DCs have a value of 1 in the dwVersion field. Windows Server 2008 and later AD DS DCs have a value of 2 in the dwVersion field. Windows Server 2003 R2 and later AD LDS DCs have a value of 2 in the **dwVersion** field.

[<50> Section 5.171](#Appendix_A_Target_50): Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 AD DS DCs have a value of 1 in the dwVersion field. Windows Server 2008 and later AD DS DCs have a value of 2 in the **dwVersion** field. Windows Server 2003 R2 and later AD LDS DCs have a value of 2 in the **dwVersion** field.

[<51> Section 5.212](#Appendix_A_Target_51): Windows Server 2008 and Windows Server 2008 R2 do not raise an ERROR\_INVALID\_PARAMETER exception when *opnum*==26 and IsAdlds() == false. Instead, the method **IDL\_DRSReplicaDemotion (section 4.1.21)** executes, and the effects vary depending on the NC specified in *pmsgIn.V1*.*pNC*.

If *pmsgIn.V1.pNC* contains the [**DSNAME**](#Section_385d478f3eb64d2cac58f25c4debdd86) of the [**default NC**](#gt_adc2c434-9679-419f-8c8b-b8c234921ad3), then:

* The return code from **IDL\_DRSReplicaDemotion** is ERROR\_SUCCESS.
* Only the [**FSMO roles**](#gt_73841222-e9d8-4dc1-83a1-206c75f4f90f) contained within the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef), as described in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.1.11, FSMO Roles, are transferred to a replication partner.
* *pmsgIn.V1.pNC*!repsFrom values are not removed.

If *pmsgIn.V1.pNC* contains the **DSNAME** of the [**config NC**](#gt_54215750-9443-4383-866c-2a95f79f1625), then:

* The return code from **IDL\_DRSReplicaDemotion** is ERROR\_INVALID\_DOMAINNAME.
* No FSMO roles are transferred.
* *pmsgIn.V1.pNC*!repsFrom values are not removed.

If *pmsgIn.V1.pNC* contains the **DSNAME** of the [**schema NC**](#gt_754c2f7e-1fe4-4d1b-8976-6dad8d13e450), then:

* The return code from **IDL\_DRSReplicaDemotion** is ERROR\_INVALID\_DOMAINNAME.
* No FSMO roles are transferred.
* *pmsgIn.V1.pNC*!repsFrom values are not removed.

If *pmsgIn.V1.pNC* contains the **DSNAME** of a domain NC and *pmsgIn.V1.pNC*!instanceType does not contain IT\_WRITE, then:

* The return code from **IDL\_DRSReplicaDemotion** is ERROR\_NO\_SUCH\_DOMAIN.
* No FSMO roles are transferred.
* *pmsgIn.V1.pNC*!repsFrom values are not removed.

If *pmsgIn.V1.pNC* contains the **DSNAME** of an [**application NC**](#gt_53fee475-b07f-45e1-b4b7-c7ac0c1e7f6a), then:

* The return code from **IDL\_DRSReplicaDemotion** is ERROR\_NO\_SUCH\_DOMAIN.
* No FSMO roles are transferred.
* *pmsgIn.V1.pNC*!repsFrom values are not removed.

# 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](mailto:dochelp@microsoft.com).

| Section | Description | Revision class |
| --- | --- | --- |
| [4.1](#Section_58f33216d9f143bfa18387e3c899c410) drsuapi RPC Interface | Updated product behavior note for this version of Windows Server. Added support information for structures DRS\_MSG\_GETCHGREQ\_V11, DRS\_MSG\_REPADD\_V3, DRS\_MSG\_REPSYNC\_V2, and DRS\_MSG\_UPDREFS\_V2. | Major |
| [4.1.10.2.1](#Section_96affbe17d93453eac759f41c0c94b3b) DRS\_MSG\_GETCHGREQ | Updated content for this version of Windows and this version of Windows Server. Added the DRS\_MSG\_GETCHGREQ\_V11 member to the DRS\_MSG\_GETCHGREQ union. | Major |
| [4.1.10.2.8](#Section_cb2bab15950b48f8af00118e186a1311) DRS\_MSG\_GETCHGREQ\_V11 | Added section with content for this version of Windows and this version of Windows Server. Added the DRS\_MSG\_GETCHGREQ\_V11 structure. | Major |
| [4.1.19.1.1](#Section_d73cb42a312c49ae82d1db93c044011e) DRS\_MSG\_REPADD | Updated content for this version of Windows and this version of Windows Server. Added the DRS\_MSG\_REPADD\_V3 member to the DRS\_MSG\_REPADD union. | Major |
| [4.1.19.1.4](#Section_63f7638bc9514486ab3f3f086d9fb622) DRS\_MSG\_REPADD\_V3 | Added section with content for this version of Windows and this version of Windows Server. Added the DRS\_MSG\_REPADD\_V3 structure. | Major |
| [4.1.23.1.1](#Section_d1ed3b4e6964468e9ef1f9a6f25cde0e) DRS\_MSG\_REPSYNC | Updated content for this version of Windows and this version of Windows Server. Added the DRS\_MSG\_REPSYNC\_V2 member to the DRS\_MSG\_REPSYNC union. | Major |
| [4.1.23.1.3](#Section_f32ab3844894416793f0eda08703d76b) DRS\_MSG\_REPSYNC\_V2 | Added section with content for this version of Windows and this version of Windows Server. Added the DRS\_MSG\_REPSYNC\_V2 structure. | Major |
| [4.1.26.1.1](#Section_2ad0252ad028412b89c92fcb7123817e) DRS\_MSG\_UPDREFS | Updated content for this version of Windows and this version of Windows Server. Added the DRS\_MSG\_UPDREFS\_V2 member to the DRS\_MSG\_UPDREFS union. | Major |
| [4.1.26.1.3](#Section_af174769feae42128e1ca6072b022e6c) DRS\_MSG\_UPDREFS\_V2 | Added section with content for this version of Windows and this version of Windows Server. Added the DRS\_MSG\_UPDREFS\_V2 structure. | Major |
| [5.39](#Section_3ee529b123db4996948a042f04998e91) DRS\_EXTENSIONS\_INT | Updated content for this version of Windows and this version of Windows Server. Added CID (DRS\_EXT\_RPC\_CORRELATIONID\_1, 0x00000400) to the dwFlagsExt bit flags. | Major |
| [5.219](#Section_589574c1eaa1456fac53de597b2cff6b) VAR\_SIZE\_BUFFER\_WITH\_VERSION | Added section with content for this version of Windows and this version of Windows Server. Added the VAR\_SIZE\_BUFFER\_WITH\_VERSION structure. | Major |
| [7](#Section_3f5d9495956344de876ace6f880e3fb2) Appendix A: Full IDL | Updated product behavior note for this version of Windows Server. Added structures DRS\_MSG\_GETCHGREQ\_V11, DRS\_MSG\_REPADD\_V3, DRS\_MSG\_REPSYNC\_V2, DRS\_MSG\_UPDREFS\_V2, and VAR\_SIZE\_BUFFER\_WITH\_VERSION. | Major |

# Index

A

Abstract types ([section 3.3.3](#section_a15fa80ed63c4ae29cd9f6df37089c89) 50, [section 3.4.3](#section_fbe9988847824858b5f25b521a44d836) 52)

[Abstract value representations](#section_284c8a5a6ede4d3488babda0b8bb59e0) 447

[Abstract value representations - converting to concrete](#section_0d7070d2f71647109f92812dc4cd8a53) 449

[AbstractPTFromConcretePT](#section_250152d133844fd3b54e3ce141a07287) 439

[AccessCheckAttr](#section_48da42bfffae4667937bd5d5627b3ea0) 439

[AccessCheckCAR](#section_4e482e032f234ee6b3b15cb367013a5d) 440

[AccessCheckObject](#section_4d2e837695a24bbfaaa9ac26c0257e4e) 440

[AccessCheckWriteToSpnAttribute](#section_f949fd9a65cf428ca9d6ffa30c2876b4) 440

[AD LDS specifics](#section_f8f70294878f4b0da368814d1f27c4b0) 60

[ADDENTRY\_REPLY\_INFO structure](#section_c4dc8d6490c04b96aa2301addd96728d) 66

[AmILHServer](#section_c3b85e98601944be96c77a59aec95526) 442

[AmIRODC](#section_b5e068cf1b68479b9c7f98f081223d25) 442

[Applicability](#section_b8a36ee2b7584b718d516cbac6c48eb6) 41

[Asynchronous processing](#section_9d615626ace2445dadfbc9189c1599be) 58

[ATRERR\_DRS\_WIRE\_V1 structure](#section_77ad67f4467045a088e7aad7a920b915) 67

[ATTR structure](#section_a2db41e278034d3ca4990fee92b1c149) 442

[ATTRBLOCK structure](#section_f81324b8640041b5bc255117589c602a) 443

[Attributes](#section_51210668de5c46afa6f23a07e3f13588) 55

[AttributeStamp](#section_2973bb80c8ed450da98159639e09820b) 443

[AttributeSyntax](#section_524cfd3b788343b78e8eb552edc9474f) 444

[AttrStamp](#section_4c58848b6a6740ac8acaab90d072fc72) 444

[ATTRTYP](#section_9117312908e6497c8266b5ac0aa5f983) 445

[AttrtypFromSchemaObj](#section_523b2d979b27407a88566a2779466701) 445

[ATTRTYP-to-OID conversion](#section_6f53317f226348ee86c14580bf97232c) 458

ATTRVAL

[abstract value representations](#section_284c8a5a6ede4d3488babda0b8bb59e0) 447

[ATTRTYP-to-OID conversion](#section_6f53317f226348ee86c14580bf97232c) 458

[concrete value representations](#section_e4816252d38c4b5f9821d23bd1dfe296) 445

[converting between abstract and concrete value representations](#section_0d7070d2f71647109f92812dc4cd8a53) 449

[overview](#section_cc002cbfefe042f89295a5a6577263d4) 445

[ATTRVAL structure](#section_cc002cbfefe042f89295a5a6577263d4) 445

[ATTRVALBLOCK](#section_b526370fdfe54e85904190d07bc16ff5) 464

[ATTRVALBLOCK structure](#section_b526370fdfe54e85904190d07bc16ff5) 464

[ATTRVALFromValue](#section_2a0d8f7e0ba448eeabc23b11d6ed2d8a) 464

B

[BOOL](#section_db93da65fa22402593309ee864c94f8f) 464

[BYTE](#section_545826e419454580961f0f0c0a47e797) 464

C

[Capability negotiation](#section_43355596b084491ba3a09176504f2987) 42

[Change tracking](#section_b8dadb6bdc5549e6b1c9afc5e91d20b4) 592

[CHANGE\_LOG\_ENTRIES](#section_b0698cac506049308b2bf88f3b5590f2) 464

[CHANGELOG\_ENTRY](#section_0861744b5ee0428fb11aa25092636b64) 465

[CheckGroupMembership](#section_a23e0a124c80485bb087ba4e989ddb59) 465

[Client initialization](#section_37b8ea496c8b45f6aa1b1125df02f0e4) 60

[ClientAuthorizationInfo](#section_92e47a548f244182b6b484f28699f8a1) 465

[ClientExtensions](#section_a2faf1f0b3dd44688b8743aec2070797) 465

Client-to-DC operations

[security](#section_19aa84e91dca485e8c85601ae845dd22) 45

[security provider](#section_a3eb3b9e73fc4896aa3acc6adc1d1523) 46

[SPN for target DC in AD DS](#section_894d09997d794e81a4077bcf6522b0a7) 46

[SPN for target DC in AD LDS](#section_3a6c821d5465414995247bec717fa60a) 47

[Common configuration example](#section_f200c87ecd674da4a45258a4761c8970) 58

[COMPRESSED\_DATA structure](#section_e4380043164748699d683c0fefa5edd7) 180

Concrete types ([section 3.3.3](#section_a15fa80ed63c4ae29cd9f6df37089c89) 50, [section 3.4.2](#section_f22a2b03808647a9af27b332301ead4d) 51)

[Concrete value representations](#section_e4816252d38c4b5f9821d23bd1dfe296) 445

[Concrete value representations - converting to abstract](#section_0d7070d2f71647109f92812dc4cd8a53) 449

[ConcretePTFromAbstractPT](#section_609de031450d4aac9c09db0ab4ccb2cb) 466

[ConfigNC](#section_0fc3974bec1f4059a0b264963aa7c2b6) 466

[Configuration example](#section_f200c87ecd674da4a45258a4761c8970) 58

[CONTREF\_DRS\_WIRE\_V1 structure](#section_9b54d7aa96a349cb8a629d326f359b83) 69

[Cookie structure](#section_2313edaf4e344b558accf8ad1593478f) 510

D

[Data display conventions](#section_5526d08bef4b48fea98da0d7813e93ed) 59

[Data types](#section_c5d9026516534ecca0d7cac691e2d08e) 439

[dc/DC](#section_c7f1df49ea514e1aa8af063ac3f5e219) 466

[DC-to-DC operations](#section_4553369b101349298ae25515f2c29c4e) 44

[DefaultNC](#section_d262eef5f2594628968041f1b799ad23) 467

[DescendantObject](#section_84e3ff511afd4b2abe081a6048158ff8) 468

[DN](#section_837c7001335148dea177c165a584816c) 468

[DNBinary](#section_b8def843aed54dcb81d99691956e6f1a) 468

[DomainNameFromNT4AccountName](#section_479ba4b06ef448b8a8161d1e8a724485) 468

[DRS\_COMP\_ALG\_TYPE enumeration](#section_bb303730066749f0b117288404c4b4cb) 180

[DRS\_COMPRESSED\_BLOB structure](#section_6d3e7f573ef846e0a6ade9331f297957) 181

[DRS\_ERROR\_DATA\_V1 structure](#section_cde11e7e4d01479c8d25400db549804e) 72

[DRS\_EXTENSIONS](#section_ed0c5dc1756648b3be084c5e26ba60c4) 468

[DRS\_EXTENSIONS structure](#section_ed0c5dc1756648b3be084c5e26ba60c4) 468

[DRS\_EXTENSIONS\_INT](#section_3ee529b123db4996948a042f04998e91) 469

[DRS\_EXTENSIONS\_INT packet](#section_3ee529b123db4996948a042f04998e91) 469

[DRS\_HANDLE](#section_55fed7de17f54ff38c53866df925056a) 472

[DRS\_MSG\_ADDCLONEDCREPLY\_V1 structure](#section_4885a24ed8014fb792dae6ad61de6d8f) 408

[DRS\_MSG\_ADDCLONEDCREQ\_V1 structure](#section_3789ff8db17942998f4d09e07c32e19d) 407

[DRS\_MSG\_ADDENTRYREPLY\_V1 structure](#section_c934ca07465d40e9a89f68f183667f1a) 65

[DRS\_MSG\_ADDENTRYREPLY\_V2 structure](#section_77568caa02be49f4b36a758776133416) 65

[DRS\_MSG\_ADDENTRYREPLY\_V3 structure](#section_1eeb493e93f1424cb8ebca74e6f051a0) 66

[DRS\_MSG\_ADDENTRYREQ\_V1 structure](#section_c02d658c9c684df5bcd5b554d7ebc873) 64

[DRS\_MSG\_ADDENTRYREQ\_V2 structure](#section_895157d524e24eaf9060148d67669e27) 64

[DRS\_MSG\_ADDENTRYREQ\_V3 structure](#section_d04ae3cc4c314c9ebb245f718a6ee646) 64

[DRS\_MSG\_ADDSIDREPLY\_V1 structure](#section_9f752d9d236d4370aadf03f2f1ecbee4) 86

[DRS\_MSG\_ADDSIDREQ\_V1 structure](#section_50b7cc92608c44ac9d3e48e2112c9bc0) 84

[DRS\_MSG\_CRACKREPLY\_V1 structure](#section_3419de890d54462e98acfb77292c91e7) 119

[DRS\_MSG\_CRACKREQ\_V1 structure](#section_b47debc059ee40e4ad0f4bc9f96043b2) 115

[DRS\_MSG\_DCINFOREPLY\_V1 structure](#section_f71a8f6c54264628aa91aeabef2c086f) 139

[DRS\_MSG\_DCINFOREPLY\_V2 structure](#section_f567e60501fe4228960e14647c29f668) 139

[DRS\_MSG\_DCINFOREPLY\_V3 structure](#section_cafc7232c6da478484d7e5d8c804c2d9) 140

[DRS\_MSG\_DCINFOREPLY\_VFFFFFFFF structure](#section_625c5133cb5b440a9f53232ae1b2dc3f) 140

[DRS\_MSG\_DCINFOREQ\_V1 structure](#section_18b23122a1c24367a677592e0d4eef18) 138

[DRS\_MSG\_EXISTREPLY\_V1 structure](#section_324d051059474c8689e838b6628bca2e) 272

[DRS\_MSG\_EXISTREQ\_V1 structure](#section_4e7acc51329a4771a6d24b490efbeb97) 271

[DRS\_MSG\_FINISH\_DEMOTIONREPLY\_V1 structure](#section_e7fee5ef4d7f459eb7c24e3b709a2f31) 156

[DRS\_MSG\_FINISH\_DEMOTIONREQ\_V1 structure](#section_687fcbf22e1f467a9a1ce2c34b3f7ce1) 155

[DRS\_MSG\_GETCHGREPLY\_V1 structure](#section_bd70a9c3c1d348cf9c24503a5567d09c) 177

[DRS\_MSG\_GETCHGREPLY\_V2 structure](#section_677d8fab6aa143279b6f62a6ad7fcfa3) 178

[DRS\_MSG\_GETCHGREPLY\_V6 structure](#section_1317a6545dd645ffaf73919cbc7fbb45) 178

[DRS\_MSG\_GETCHGREPLY\_V7 structure](#section_26eaca610f1947e7b3042580e9870aa8) 179

[DRS\_MSG\_GETCHGREPLY\_V9 structure](#section_b9564a194500444ba99b0da1b08cdb6f) 179

[DRS\_MSG\_GETCHGREQ\_V10 structure](#section_92b1b77d205846e09e8c6664b96a0cf9) 175

[DRS\_MSG\_GETCHGREQ\_V3 structure](#section_6a2a056cac7f47d09e6d9023a4e5947c) 171

[DRS\_MSG\_GETCHGREQ\_V4 structure](#section_9db4db218ccd4c8186626e2baff8426c) 172

[DRS\_MSG\_GETCHGREQ\_V5 structure](#section_fd24b73c7b8143af8c7765bc2e3181b7) 173

[DRS\_MSG\_GETCHGREQ\_V7 structure](#section_5ef4f597a3974f6fa98b7a034247d886) 173

[DRS\_MSG\_GETCHGREQ\_V8 structure](#section_4304bb4ae9b54c8a8731df4d6f9ab567) 174

[DRS\_MSG\_GETMEMBERSHIPS2\_REPLY\_V1 structure](#section_92cb5c90905a4142acd5c79ba4ac6872) 168

[DRS\_MSG\_GETMEMBERSHIPS2\_REQ\_V1 structure](#section_91151fe919eb40118d49ae38e4215c3e) 167

[DRS\_MSG\_GETREPLINFO\_REQ\_V1 structure](#section_8437a79c4d0046a1b13ea7f97d2f6608) 276

[DRS\_MSG\_GETREPLINFO\_REQ\_V2 structure](#section_a5f01efdecd942c6a0c78cf67d028589) 276

[DRS\_MSG\_INIT\_DEMOTIONREPLY\_V1 structure](#section_5c5340e1e9844d3e84ec3e2efa555366) 323

[DRS\_MSG\_INIT\_DEMOTIONREQ\_V1 structure](#section_d71cd7834d7b443cadfe6b3a77a19671) 322

[DRS\_MSG\_KCC\_EXECUTE\_V1 structure](#section_8111f93307f44c59849750c0f8970eb5) 152

[DRS\_MSG\_MOVEREPLY\_V1 structure](#section_7a469f429d324cacb6d729c1ad175ab0) 326

[DRS\_MSG\_MOVEREPLY\_V2 structure](#section_59b23876f18c44bdb88f3848d256c61e) 326

[DRS\_MSG\_MOVEREQ\_V1 structure](#section_ea7581961fbe48d691e7f1a3105fff9a) 325

[DRS\_MSG\_MOVEREQ\_V2 structure](#section_d77d56704a3549d2b6c8e314a7f76ff3) 325

[DRS\_MSG\_NT4\_CHGLOG\_REPLY\_V1 structure](#section_1bc2090ed3ec443588b902437207ea22) 265

[DRS\_MSG\_NT4\_CHGLOG\_REQ\_V1 structure](#section_27207d2f7de4465882dd228bfea28c49) 264

[DRS\_MSG\_QUERYSITESREPLY\_V1 structure](#section_1fdb59889d5c47fda917c579360f6188) 336

[DRS\_MSG\_QUERYSITESREPLYELEMENT\_V1 structure](#section_1a57b4d2004d4843962a0884abfb00c6) 337

[DRS\_MSG\_QUERYSITESREQ\_V1 structure](#section_a5e57fa8944144c6af986437db28d6d6) 336

[DRS\_MSG\_REPADD\_V1 structure](#section_7dc022f441a54d7882f49db6e60454b9) 363

[DRS\_MSG\_REPADD\_V2 structure](#section_892d85776e1545ba8e7a022807ed8649) 363

[DRS\_MSG\_REPDEL\_V1 structure](#section_fe3b5d94ff38448182193d1e38d9fc40) 368

[DRS\_MSG\_REPLICA\_DEMOTIONREPLY\_V1 structure](#section_33b626958d9d4a258d37320131470cca) 373

[DRS\_MSG\_REPLICA\_DEMOTIONREQ\_V1 structure](#section_8e459d5d129e40be8bc90872f763c61d) 372

[DRS\_MSG\_REPMOD\_V1 structure](#section_34626189925f478693083d65ae3ce2ac) 376

[DRS\_MSG\_REPSYNC\_V1 structure](#section_d29519a5f85e4bd5907a0777ce0be29f) 379

[DRS\_MSG\_REPVERIFYOBJ\_V1 structure](#section_b81dc433007c4a0eae27a0469cc25c58) 383

[DRS\_MSG\_REVMEMB\_REPLY\_V1 structure](#section_a8a132988e3b408dbad76710138c54cd) 161

[DRS\_MSG\_REVMEMB\_REQ\_V1 structure](#section_bc96a03b579e44548db412067b6ca985) 160

[DRS\_MSG\_RMDMNREPLY\_V1 structure](#section_ed5d05b9726449fa8cc4d393791eec86) 356

[DRS\_MSG\_RMDMNREQ\_V1 structure](#section_f7f04091302e470fae477244192fe599) 355

[DRS\_MSG\_RMSVRREPLY\_V1 structure](#section_f567b19b534943228e740925203ee682) 359

[DRS\_MSG\_RMSVRREQ\_V1 structure](#section_05843a968bf74c08b52d5c472ed04a1a) 358

[DRS\_MSG\_SPNREPLY\_V1 structure](#section_23f5b13002ba48c2a2b5cafff417aafc) 403

[DRS\_MSG\_SPNREQ\_V1 structure](#section_463869d728a54e158c9b0e3376e9acff) 403

[DRS\_MSG\_UPDREFS\_V1 structure](#section_ee70be3308dc48b2a59943a50943c0e1) 391

[DRS\_MSG\_VERIFYREPLY\_V1 structure](#section_d675aba6c9524e1088693243408756f4) 397

[DRS\_MSG\_VERIFYREQ\_V1 structure](#section_4593a76b71f14e4cb5e14d2e27afb3cb) 396

[DRS\_OPTIONS](#section_ac9c8a11cd464080acbf9faa86344030) 473

[DRS\_SecBuffer](#section_a5ad6879cf0c411c96c32eb7ea37149c) 475

[DRS\_SecBuffer structure](#section_a5ad6879cf0c411c96c32eb7ea37149c) 475

[DRS\_SecBufferDesc](#section_aa6f5b36cf1e49c9b5fd4cd5c9de7448) 477

[DRS\_SecBufferDesc structure](#section_aa6f5b36cf1e49c9b5fd4cd5c9de7448) 477

[DRS\_SPN\_CLASS](#section_3e0e2ef55dc54b37a927323f3c4876ca) 477

[DS\_DOMAIN\_CONTROLLER\_INFO\_1W structure](#section_b30c5951ccb14fb6ba9a5699d5d78759) 140

[DS\_DOMAIN\_CONTROLLER\_INFO\_2W structure](#section_a9c9fd5024b54ff7b3368e23ac0622de) 141

[DS\_DOMAIN\_CONTROLLER\_INFO\_3W structure](#section_08f99ee78235482bbfe5c6170f133cd4) 142

[DS\_DOMAIN\_CONTROLLER\_INFO\_FFFFFFFFW structure](#section_38259d4611e64e748e0c0b0f9ce2dab4) 142

[DS\_NAME\_FORMAT enumeration](#section_73c73cf208244d6597f4f56244f3e8a6) 117

[DS\_NAME\_RESULT\_ITEMW structure](#section_e174fead5a374a11a0f669086e8dd4e9) 118

[DS\_NAME\_RESULTW structure](#section_0076d2413f794b0b8e078ccfaff8bd4c) 118

[DS\_REPL\_ATTR\_META\_DATA structure](#section_f7a10e539c454719a6414db15e385297) 283

[DS\_REPL\_ATTR\_META\_DATA\_2 structure](#section_8036e127f8ae4341b21886db427d5791) 284

[DS\_REPL\_ATTR\_VALUE\_META\_DATA structure](#section_8e3dbb537f7f4c379095921fa9c0a4df) 286

[DS\_REPL\_ATTR\_VALUE\_META\_DATA\_2 structure](#section_78196358ad2a4c6b83ffcef40577ba85) 287

[DS\_REPL\_CLIENT\_CONTEXT structure](#section_a540a2eb228b447aba6f6868d7793c53) 288

[DS\_REPL\_CLIENT\_CONTEXTS structure](#section_7fe11aa454ca4d618ab6bd0f51e8c9e9) 288

[DS\_REPL\_CURSOR structure](#section_cf960f2fc8fa4dfa9252f70164c14039) 281

[DS\_REPL\_CURSOR\_2 structure](#section_40366a5b9a48465fb7ac03f56334f76d) 281

[DS\_REPL\_CURSOR\_3W structure](#section_6c7bd13e06b8459a87fce67c7954290d) 282

[DS\_REPL\_CURSORS structure](#section_bfab2029039c442e8a924378d3a27473) 280

[DS\_REPL\_CURSORS\_2 structure](#section_e246043af2084058a00fa8b19e479dc3) 281

[DS\_REPL\_CURSORS\_3W structure](#section_63ecf5a190424f64b2c4ba94f9b13064) 282

[DS\_REPL\_KCC\_DSA\_FAILURESW structure](#section_ecdca8cec07f4bd5839770b610bed8fc) 284

[DS\_REPL\_KCC\_DSA\_FAILUREW structure](#section_5d5ac3d8dc80401b9ca81e7a385024a4) 284

[DS\_REPL\_NEIGHBORSW structure](#section_c0f48bab491a4d6585b04045fc721bb4) 279

[DS\_REPL\_NEIGHBORW structure](#section_ab63bcfb34cf4d4cb201f43380c3d745) 279

[DS\_REPL\_OBJ\_META\_DATA structure](#section_a2ce73faaacd49d7911110ef08001963) 282

[DS\_REPL\_OBJ\_META\_DATA\_2 structure](#section_37c78898e01f4e279e186f357bba744f) 283

[DS\_REPL\_OP\_TYPE](#section_bf047cfe32bd43f693d3b67b05eaac66) 477

[DS\_REPL\_OP\_TYPE enumeration](#section_bf047cfe32bd43f693d3b67b05eaac66) 477

[DS\_REPL\_OPW structure](#section_9c0d0bd127bb44a38317969fe45e87e1) 285

[DS\_REPL\_PENDING\_OPSW structure](#section_cce01b0ae66f4ed2992d39a69e480926) 285

[DS\_REPL\_SERVER\_OUTGOING\_CALL structure](#section_8e769e6aa8b24291aeff1ec521c96852) 289

[DS\_REPL\_SERVER\_OUTGOING\_CALLS structure](#section_4a8c35811f9448f58c02f2d2ec6cef46) 289

[DS\_REPL\_VALUE\_META\_DATA structure](#section_8e53006b9e1d48e6ba5fc675c0a98b3a) 286

[DS\_REPL\_VALUE\_META\_DATA\_2 structure](#section_ddf21c4e616b46798822e1879ee05f6e) 287

[DSA\_ADDRESS\_LIST\_DRS\_WIRE\_V1 structure](#section_2f6ca421320d4fee96b656315c446236) 69

[DSA\_MSG\_EXECUTE\_SCRIPT\_REPLY\_V1 structure](#section_84fef2b202bb4832b0d1b7eca9689e76) 436

[DSA\_MSG\_EXECUTE\_SCRIPT\_REQ\_V1 structure](#section_46742a7cb8594e4ea64f58d572b3bde8) 436

[DSA\_MSG\_PREPARE\_SCRIPT\_REPLY\_V1 structure](#section_1c046d5461674da8b02863ec9a235b6f) 432

[DSA\_MSG\_PREPARE\_SCRIPT\_REQ\_V1 structure](#section_8664f791fd1e4f6caa2ab0703d0786a0) 431

[DSA\_RPC\_INST](#section_88a396196dbe4ba184355966c1a490a7) 478

[DSA\_RPC\_INST structure](#section_88a396196dbe4ba184355966c1a490a7) 478

[DSAObj](#section_cab719ea13d34c49b68162dd438a97d4) 477

DSName ([section 5.49](#section_a0d5477a522946b9890a54b924d487d1) 478, [section 5.50](#section_385d478f3eb64d2cac58f25c4debdd86) 478)

[DSNAME equality](#section_73376a9ab03e4655a8d052451b7ec74b) 480

[DSNAME structure](#section_385d478f3eb64d2cac58f25c4debdd86) 478

[DSTIME](#section_a72a16b973e441caa5c1afc5fc54e175) 480

[DWORD](#section_60c3f5f194924d1083c89a155e162ef3) 481

E

[ENCRYPTED\_PAYLOAD packet](#section_7b60d2b35bb149aaaefcfa887e683977) 181

[ENTINF](#section_6d69822eadb649778553c2d529c17e5b) 481

[ENTINF structure](#section_6d69822eadb649778553c2d529c17e5b) 481

[ENTINF\_EnumerateAttributes](#section_cfbf6afdc2c946cc8772817dd7591bd8) 482

[ENTINF\_GetValue](#section_9ccbd0eb3cd740baa49df13accce7442) 481

[ENTINF\_SetValue](#section_b74eb83d96ed45389020413070311548) 482

[ENTINFLIST](#section_59cad1ffe499477c8c9e59939a71aff5) 482

[ENTINFLIST structure](#section_59cad1ffe499477c8c9e59939a71aff5) 482

Examples

[common configuration example](#section_f200c87ecd674da4a45258a4761c8970) 58

[data display conventions](#section_5526d08bef4b48fea98da0d7813e93ed) 59

[IDL\_DRSBind method example](#section_26f0f2f4ec9342a48f96e2bd63294e71) 111

[Expunge](#section_b4dfb24533e244eba5e16419d74907ca) 483

F

[Fields - vendor-extensible](#section_9b545b76247c44aaba77a3581fa4205a) 42

[FILETIME](#section_70ee934bc9b944498aa36dfe9cef3eff) 483

[FilteredGCPAS](#section_ea98c9b838df4453bb41f443fcf42faf) 483

[FilteredPAS](#section_af73c5d1b71f409da508435a247bd876) 483

[FindChar](#section_48e97e384022421583d360e1c4c40603) 484

[FindCharRev](#section_b9aebe8fa9fd46299df2cb286f803df5) 484

[FOREST\_TRUST\_INFORMATION](#section_642c0d174f3b4752a9a3464b080bf0b6) 485

[FOREST\_TRUST\_INFORMATION packet](#section_642c0d174f3b4752a9a3464b080bf0b6) 485

[FOREST\_TRUST\_RECORD\_TYPE](#section_5514b72b8452446aaa64abb35536baca) 494

[FOREST\_TRUST\_RECORD\_TYPE enumeration](#section_5514b72b8452446aaa64abb35536baca) 494

[ForestRootDomainNC](#section_e9b76fbbe90c4e5ea77da8e21a0717eb) 495

[Full IDL](#section_3f5d9495956344de876ace6f880e3fb2) 557

[FullReplicaExists](#section_55dc0083c1ba481da0496fb20e3bbb56) 495

G

[GCPAS](#section_cdd3356b137f4ba68128e29ba330db75) 495

[GetAttrVals](#section_a604aaabcc4245b2a62cb2beb71aae86) 497

[GetCallerAuthzInfo](#section_d7bdf561886f4637bc3fd5fe18250689) 497

[GetDefaultObjectCategory](#section_04943165661f4f72ab328f6802b00a17) 497

[GetDSNameFromDN](#section_6a86b787444540c0aff5d6119c11e22a) 498

[GetDSNameOfEnterpriseRODCsGroup](#section_eb9ce9e775ab4efbaac484f883749d67) 163

[GetFilteredAttributeSet](#section_4094b3e8061a49b28ffb3d6d55922718) 495

[GetForestFunctionalLevel](#section_d90dd8c214254e4692f58d93e812d811) 498

[GetFSMORoleOwner](#section_73171ad384bc455dae6837b66a920789) 498

[GetInstanceNameFromSPN](#section_c456de965cbf4915a5d6dfa94163a796) 499

[GetNCType](#section_4f17dcc04ce14653bfa72b841b7966b3) 496

[GetObjectNC](#section_7400e082479b454f8f76c6aaae9c0a49) 499

[GetProxyEpoch](#section_806dc0cb3bfa4c379f5e331764118012) 499

[GetProxyType](#section_61229f40423b42c78be6e3c9c515f7e6) 499

[GetServiceClassFromSPN](#section_811b10c06b754998ab7676880796abee) 499

[GetServiceNameFromSPN](#section_ed78e7d4a02d448c8469cd7f4ae3558e) 500

[Glossary](#section_e5c2026bf7324c9d9d60b945c0ab54eb) 23

[groupType bit flags](#section_da82c296fcc74597b1824a84e40fb169) 500

[GUID](#section_5e740f50e6a048c9bca800072e85d963) 500

[GuidFromString](#section_2bba7154833944c0bf18776e3e7cb15c) 501

[GuidToString](#section_90300e394b9c413ea6170cec5ad83e8b) 501

H

[handle\_t](#section_4fe8cfe5f4014d88b141a0a23260a15b) 501

I

[IDL](#section_3f5d9495956344de876ace6f880e3fb2) 557

[IDL\_DRSAddCloneDC method](#section_ef0bfb1d037b4626a6d9cc7589bc5786) 406

[IDL\_DRSAddEntry method](#section_06764fc54df64104b6afa92bdaa81f6e) 63

[IDL\_DRSAddSidHistory method](#section_376230a5d8064ae5970af6243ee193c8) 83

[IDL\_DRSBind method](#section_605b1ea19cdc428fab7a70120e020a3d) 101

[IDL\_DRSBind method example](#section_26f0f2f4ec9342a48f96e2bd63294e71) 111

[IDL\_DRSCrackNames method](#section_9b4bfb4466564404bcc8dc88111658b3) 114

[IDL\_DRSDomainControllerInfo method](#section_668abdc81db741049deafeab05ff1736) 138

[IDL\_DRSExecuteKCC method](#section_ad807917687b40d9abe2053af0246523) 152

[IDL\_DRSFinishDemotion method](#section_0bf530e81be04f48b8c2208031a8725f) 154

[IDL\_DRSGetMemberships method](#section_d5ace4527cdd4d50bb6439b4c55180a2) 159

[IDL\_DRSGetMemberships2 method](#section_d4e67cc32ee14b2b8055cebefc556252) 166

[IDL\_DRSGetNCChanges method](#section_b63730ac614c431c950128d6aca91894) 168

[IDL\_DRSGetNT4ChangeLog method](#section_6e000eb660fd4d6cae82bb6479df02fa) 263

[IDL\_DRSGetObjectExistence method](#section_6355d4f5f5564527adde37afba2fcf56) 270

[IDL\_DRSGetReplInfo method](#section_dd29f9ceb30b411ebd54b77634eded47) 275

[IDL\_DRSInitDemotion method](#section_faca17da3f7f409298dbfd2ce7d98b8c) 322

[IDL\_DRSInterDomainMove method](#section_595b2fef493b4b1db60de7a1a3345b0e) 324

[IDL\_DRSQuerySitesByCost method](#section_2c3faba2d64e4866b8f1fc8d5f4ec710) 335

[IDL\_DRSRemoveDsDomain method](#section_aa3cfa46c737425aae65ecaf9efe7e84) 354

[IDL\_DRSRemoveDsServer method](#section_d5c310ae347a49d4a78e6ffb2eecd581) 357

[IDL\_DRSReplicaAdd method](#section_7219df914eea494f88e3780d40d2d559) 362

[IDL\_DRSReplicaDel method](#section_1420a9bf9267464da6d57676472d7f1d) 367

[IDL\_DRSReplicaDemotion method](#section_8a2f0388bdfb4519a8c3384f27c11639) 371

[IDL\_DRSReplicaModify method](#section_cd241bf256be453786b1cdbc997b0860) 376

[IDL\_DRSReplicaSync method](#section_25c71d91051f4c26977fa70892f29b00) 379

[IDL\_DRSReplicaVerifyObjects method](#section_8dba150d50f647f1941e1a606c30db0b) 382

[IDL\_DRSUnbind method](#section_49eb17c9b6a94ceabef866abda8a7850) 389

[IDL\_DRSUpdateRefs method](#section_a273bbcfaeca46088ad4127d3e597cd4) 390

[IDL\_DRSVerifyNames method](#section_80739a29e8ed44788490475a18e9779d) 396

[IDL\_DRSWriteSPN method](#section_8b129dc8ed4545379555b6fef764ab7d) 402

[IDL\_DSAExecuteScript method](#section_1cb59761aeae4f448f9e06ae75ae45ca) 435

[IDL\_DSAPrepareScript method](#section_749197848e574cf5840f6f1bd226cf02) 431

[Implementer - security considerations](#section_d0f3c777575241529f1b2b7818f2ebdf) 556

[Index of security parameters](#section_15eaf1fae81f4e88b1470241b9c86193) 556

[Informative references](#section_316590535581441d84adbec58b53a3f5) 37

[Initialization](#section_37b8ea496c8b45f6aa1b1125df02f0e4) 60

[instanceType bit flags](#section_5e821e19c93c4e619cf1453d6bdfec56) 501

[INT32 packet](#section_4160557351bc49c2abcdd8d4c31ab1f5) 446

[INT64 packet](#section_1c3855efb0584248866f70aa740b5a7b) 446

[INTFORMPROB\_DRS\_WIRE\_V1 structure](#section_bdfbc428fa77476788bdca75750b03bf) 68

[Introduction](#section_06205d9730da4fdca2763fd831b272e0) 22

[Is2PartSPN](#section_c8c26ce1d9a74e10b1bb191a70413228) 502

[Is3PartSPN](#section_e6177bb2bf394eea862c0951cdd0dac6) 502

[IsBuiltinPrincipal](#section_5e95f55d0f894f21989df2a375556eac) 502

[IsDCAccount](#section_e6f1a43b613c43279a57b61a82dec535) 503

[IsDomainNameInTrustedForest](#section_6d30d09105774ec8813f48aa2d99e068) 502

[IsGC](#section_51d97a3468c848afbd94c800d085f7ce) 503

[IsGetNCChangesPermissionGranted](#section_15bbda5277424ce39327018647b0fc26) 503

[IsGUIDBasedDNSName](#section_a7cf8c3f1f9645feb6c1546c839605ca) 504

[IsMemberOfBuiltinAdminGroup](#section_d9e36aa86caf46a5a90750b64f77bc99) 504

[IsRecycleBinEnabled](#section_d3b938b758b147869a493726a77874ca) 504

[IsRevealFilteredAttribute](#section_84e5f1d6e2d7494db232222280a721fb) 504

[IsRevealSecretRequest](#section_e72853a19b9c4931b76aa417581890f7) 505

[IsValidServiceName](#section_9ff2b154b44d4b25833c26381b3ae6ca) 506

K

[KCCFailedConnections](#section_eaffa80d8baf4784898ee9fbc7bd8296) 507

[KCCFailedLinks](#section_fec285f37f034cfc89ac911f61c0c7d3) 507

L

Language constructs ([section 3.4.2](#section_f22a2b03808647a9af27b332301ead4d) 51, [section 3.4.3](#section_fbe9988847824858b5f25b521a44d836) 52, [section 3.4.4](#section_3f796d05597445e1bc8ff06cc3bce6bb) 54)

[LARGE\_INTEGER](#section_ebf2c36755e84066b879f80f1e8f69a9) 507

[LDAP\_CONN\_PROPERTIES](#section_09a9cd41caed441da7515a992800a4fb) 507

LDAP\_SERVER\_DIRSYNC\_OID LDAP search control

[abstract types](#section_17b6e94e511247cf885859553abf2c5a) 508

[concrete types](#section_2ba46be555e44e42bccb76058ac858dd) 510

[DirSyncReqToGetChgReq](#section_2f4b9020fc524a83b5d5b0c5ad141c5c) 512

[GetChgReplyToSearchResult](#section_263edd41c18744ebbafdb86b7b2bbde2) 513

[GetResponseDirSyncControlValue](#section_13caba351778481db64eabc3393cb8f6) 516

[GetUsnUtdVectorFromCookie](#section_98781a8bb51e4bc7b43fce5855866fa2) 517

[IsFilteredAttributePresent](#section_6a247f4db65540be9c5147c10de9bd8c) 519

[LDAPDisplayNameFromAttrTyp](#section_75fba0eb60b344489da3352bf5b672c6) 516

[overview](#section_c9fa9860cf2f4877b7709e4aa73c9c5b) 508

[ProcessDirSyncSearchRequest](#section_0e61ce24eac740d1a01437854e932b43) 511

[SecurityCheckForChanges](#section_4abd4c7cc078480995bc38e657d5c034) 518

[TransformDSNameToLdapDN](#section_c2d53ab0a8f64348997376df4238c99d) 516

[TransformEntinfToSearchEntry](#section_dfa0142e78d14294b4631b33a92cd8b0) 514

[TransformReplValInfListToSearchEntry](#section_bad68e9bf4e64d9ea9c77db72ef220bf) 515

[LDAPConnections](#section_3653bad8fb4d4edda52179b19c1c1e95) 519

[LinkStamp](#section_85075a8ff8a848d390e633ca806e31a7) 520

[LinkValueStamp](#section_6a9517897afa47dda96c83fc0e30aa3d) 520

[LocalAttidFromRemoteAttid](#section_5e30bd01b1dd4019b4061e9c2472f359) 522

[LONG](#section_0fdb03d734b44921b9c46e8025f9e795) 522

[LONGLONG](#section_779543e505e340fa84c24f96f919feff) 522

[LPWSTR](#section_fd9d8705332d4df5858fec4959de3639) 522

M

[MakeAttid](#section_3f7127da084a4cb2ad1349871a733c91) 522

[MakeProxyValue](#section_0122b9a5ddac4d548ef80fd039aebfb8) 522

[MasterReplicaExists](#section_05a6d0569d0d4181a48105cb2d6bb61e) 522

[MD5\_CTX](#section_c5c96a8ec367481089afb51119c7e0e7) 523

[MD5Final](#section_b58b0ff5198d4c9496efa3e557743b30) 523

[MD5Init](#section_e463b79e548f4127a1bb059adf6fdf38) 523

[MD5Update](#section_0651733f1b3f4eb6a4d145a161631776) 523

[MergeUTD](#section_1f46eb56c4bc4e68917466458ae4bb9b) 523

Messages

[overview](#section_d8a5de321b81441cb3bcce90b1ccd178) 43

[transport](#section_e6076eab53f64aad9041888c5734b715) 43

Methods ([section 1.3.1](#section_beebc62bcceb4705aafe14fa53fa19f4) 37, [section 4](#section_9554afa5e7554742a34b899fc4e2fd20) 61)

[MTX\_ADDR](#section_107b7c0e0f0d4fe2823214ec3b78f40d) 523

[MTX\_ADDR structure](#section_107b7c0e0f0d4fe2823214ec3b78f40d) 523

N

[NAMERESOP\_DRS\_WIRE\_V1 structure](#section_5426b9b7ba824f1ba58abfafb8c12ba6) 69

[NAMERR\_DRS\_WIRE\_V1 structure](#section_b1d8c71ef3684394a35612e6f23a5eca) 68

[Naming conventions](#section_ecd3aa7edf3c4ba4849f5dfbcdfd8ec2) 51

[NCType bits](#section_9cdea66549714fca92b57558793bf11c) 524

[NetworkAddress](#section_3d0d3777935847ddb55534405f57f912) 524

[NewPrefixTable](#section_e1830191fedd422690de014d0fa73a26) 525

[Normative references](#section_0d18839cdfb0413b975030b1f95aeff6) 35

[NT4\_REPLICATION\_STATE structure](#section_a7d8f9243fc04dc189f359f4521cf0e7) 266

[Nt4ReplicationState](#section_4ee4212d910042aeab10c480bcce73a9) 525

[NT4SID](#section_8fb66015d04947b9804d1372b1afc9fc) 525

[NT4SID structure](#section_8fb66015d04947b9804d1372b1afc9fc) 525

[NTDSTRANSPORT\_OPT values](#section_e3f248d9bed94b41b4dfefcbe14aa957) 526

[NULLGUID](#section_61ffe2b7006f4bdeaf6cc3ecb8998eb9) 526

O

[Object attributes](#section_51210668de5c46afa6f23a07e3f13588) 55

[Object(Access-Point)](#section_7bf069fd100640ac8707d7e7d34a1de0) 448

[Object(DN-Binary)](#section_53e0cab126874bb1984b8063240e7430) 448

[Object(DN-String)](#section_de1cb4d35bb04ff59da8937a7dd1134a) 448

[Object(DS-DN)](#section_32896221c4cb4f2ca6a7bc2cece8fc7b) 447

[Object(OR-Name)](#section_c030a604f05f4c6fb74259c8607e7c12) 449

[ObjExists](#section_4df4a05a7092466984529b9922d07e58) 526

[OID](#section_339504853a964b668a28a3a33e80302b) 526

[OID\_t](#section_cbc2b76189384591a9f72d1512ed7f05) 526

[OID\_t structure](#section_cbc2b76189384591a9f72d1512ed7f05) 526

[OidFromAttid](#section_fc3acd51af2a41e085977037e01454cc) 526

[Organization](#section_41fbd822823a4121b93ee037c420c189) 49

[Overview](#section_c56432ffaf884443b500eecb3047da4d) 49

[Overview (synopsis)](#section_86d291eafe5c48fdacca82649feb5654) 37

P

[Parameters - security index](#section_15eaf1fae81f4e88b1470241b9c86193) 556

[parent](#section_a5c8f9655943456a810140161c8deeed) 527

[PARTIAL\_ATTR\_VECTOR\_V1\_EXT](#section_1d5c1b34daa44761a8b5d3c0146a0e30) 527

[PARTIAL\_ATTR\_VECTOR\_V1\_EXT structure](#section_1d5c1b34daa44761a8b5d3c0146a0e30) 527

[partialAttributeSet](#section_648bae447db64d26af5830ff1b65bd7a) 527

[PartialGCReplicaExists](#section_cd801adced8f4965ac091866e4e0572e) 527

[PAS\_DATA](#section_14e0e4828f1d4fa589ded9fd7f98b10d) 527

[PAS\_DATA packet](#section_14e0e4828f1d4fa589ded9fd7f98b10d) 527

[PdcChangeLog](#section_24444ebf2e164050af0890c85e611234) 528

[PDS\_NAME\_RESULT\_ITEMW](#section_e174fead5a374a11a0f669086e8dd4e9) 118

[PDS\_NAME\_RESULTW](#section_0076d2413f794b0b8e078ccfaff8bd4c) 118

[PDSA\_RPC\_INST](#section_88a396196dbe4ba184355966c1a490a7) 478

[PerformAddOperation](#section_e61c9ac3cd7c47d3ab1b0c8b61dc4869) 528

[PerformAddOperationAsSystem](#section_2b75b1b502c942b48e926b154bd2893b) 529

[PerformModifyOperation](#section_ca4c645fce37426080b4f1555e644ef3) 232

[Preconditions](#section_c3379fc5123d4f28922d97308561d748) 41

[PrefixTable](#section_2789d96b50e8444d82d6523831556d76) 529

[PrefixTableEntry](#section_d26d36cd10c44b27a84e98336abf357a) 529

[PrefixTableEntry structure](#section_d26d36cd10c44b27a84e98336abf357a) 529

Prerequisites ([section 1.5](#section_c3379fc5123d4f28922d97308561d748) 41, [section 3.3.1](#section_31b053e843244812950d0619793d64a6) 49)

[PROBLEMLIST\_DRS\_WIRE\_V1 structure](#section_96ee7875263b4fc79fa31dc7d19c8d9d) 67

[Procedures](#section_c5d9026516534ecca0d7cac691e2d08e) 439

[Processing - asynchronous](#section_9d615626ace2445dadfbc9189c1599be) 58

[Product behavior](#section_e3b895564e48467c87b3b4a737cc4999) 584

[PROPERTY\_META\_DATA structure](#section_ab1ad92035384d6491976e700ef1f222) 183

[PROPERTY\_META\_DATA\_EXT](#section_aef7ebdec305422495fd585c86b19c38) 529

[PROPERTY\_META\_DATA\_EXT structure](#section_aef7ebdec305422495fd585c86b19c38) 529

[PROPERTY\_META\_DATA\_EXT\_VECTOR](#section_22bccd511e7d4502aef8b84da983f94f) 530

[PROPERTY\_META\_DATA\_EXT\_VECTOR structure](#section_22bccd511e7d4502aef8b84da983f94f) 530

[proxiedObjectName value format](#section_0da1cf6f630a4fd5939f757b950d63b0) 530

[Pseudocode](#section_887dc359a000459f806f82f8a06eb9cc) 51

R

[RDN](#section_042a097c68374e16b04fdd81e65f6613) 530

[rdnType](#section_3f1dbfeafcde4197ace86ede31cf46e7) 530

[Record packet](#section_2a16b808322f433ab5a671eefba82b5a) 485

[RecycleObj](#section_4ab30c288ced4ab5b1f40a9146641515) 530

References

[informative](#section_316590535581441d84adbec58b53a3f5) 37

[normative](#section_0d18839cdfb0413b975030b1f95aeff6) 35

[REFERR\_DRS\_WIRE\_V1 structure](#section_5d52fff985c54407955ddbf5630ec37b) 68

[Relationship to other protocols](#section_0597aff601774d5299f214a5441bc3c1) 41

[RemoveObj](#section_2ab999f278fc4bfa97e1c148df3ec177) 531

[REPLENTINFLIST](#section_c38b0412cf004b0cb4f44662a4484a00) 531

[REPLENTINFLIST structure](#section_c38b0412cf004b0cb4f44662a4484a00) 531

[ReplicationQueue](#section_6226aaa1178d45ff9e17815556739595) 532

[REPLTIMES](#section_42d7d8e8794e427998028b5916e8b099) 532

[REPLTIMES structure](#section_42d7d8e8794e427998028b5916e8b099) 532

[replUpToDateVector/ReplUpToDateVector](#section_8cb40d62a51d47e39b4e0837edffd61c) 533

[REPLVALINF](#section_22946fbf170e4ab482c7dabdfd97bf5a) 533

[REPLVALINF structure](#section_22946fbf170e4ab482c7dabdfd97bf5a) 533

[REPS\_FROM](#section_f8e930ead84745858d58993e05f55e45) 534

[REPS\_FROM packet](#section_f8e930ead84745858d58993e05f55e45) 534

[REPS\_TO](#section_b422aa877d074527b070c5d719696c43) 536

[REPS\_TO packet](#section_b422aa877d074527b070c5d719696c43) 536

[repsFrom/RepsFrom](#section_3ef27d3cb9c944048e53ebf3a64a9a10) 539

[repsTo/RepsTo](#section_302391a9f6e14c0ca1b25604a42e982b) 540

[RevealSecretsPolicy enumeration](#section_dff72b445a8545b8bc9ba6faab723610) 214

[REVERSE\_MEMBERSHIP\_OPERATION\_TYPE enumeration](#section_66e09464e14347e48626f10772ee6882) 160

[Rid](#section_bfa9c9d4ef344a7abb8353dda81700a1) 542

[Right](#section_6c6bd72877e142be9a0805e8c21e3253) 542

[RIGHT values](#section_a0a54f6a63844122bd0b29a782f213b2) 542

[RPCClientContexts](#section_65d838f52f694c228b263340182dcde1) 542

[RPCOutgoingContexts](#section_b9f465938a4041869ecebc2612b4c3f4) 543

S

[sAMAccountType values](#section_cc44dc4cfb884ffb9a29148fb39002d4) 543

[SCHEMA\_PREFIX\_TABLE](#section_9b371267e8b84c69997902dae02e5e38) 544

[SCHEMA\_PREFIX\_TABLE structure](#section_9b371267e8b84c69997902dae02e5e38) 544

[SchemaNC](#section_ea66faf965614f49bddf264389df05e4) 544

[SchemaObj](#section_690ac1caecb8480c9f38c37b5a19380f) 544

[SECERR\_DRS\_WIRE\_V1 structure](#section_3f4e4f6956524318b898a22d5907403c) 70

Security

[background](#section_d447826c54ff46bd8d35b0c0644e9e95) 43

[client-to-DC operations](#section_19aa84e91dca485e8c85601ae845dd22) 45

[DC-to-DC operations](#section_4553369b101349298ae25515f2c29c4e) 44

[implementer considerations](#section_d0f3c777575241529f1b2b7818f2ebdf) 556

[overview](#section_0a156712918047bab0802e285f127a7f) 43

[parameter index](#section_15eaf1fae81f4e88b1470241b9c86193) 556

provider ([section 2.2.3.1](#section_2fe42873129f4f10a0fa2e2360fe6ce9) 44, [section 2.2.4.1](#section_a3eb3b9e73fc4896aa3acc6adc1d1523) 46)

[service authentication](#section_599d1fe805494653bdcecb51965c19ae) 43

[SPN for target DC in AD DS](#section_894d09997d794e81a4077bcf6522b0a7) 46

[SPN for target DC in AD LDS](#section_3a6c821d5465414995247bec717fa60a) 47

[Security provider](#section_2fe42873129f4f10a0fa2e2360fe6ce9) 44

[Sequencing issues](#section_67c5a415a6c740988cf36ef8d173cfe8) 38

[Server extensions](#section_88a71c37b18d4918a0ebb532351489f5) 544

[Server initialization](#section_37b8ea496c8b45f6aa1b1125df02f0e4) 60

[Service authentication](#section_599d1fe805494653bdcecb51965c19ae) 43

[SID](#section_13560cc227ff43a09d6fd686bccc5f3c) 545

[SidFromStringSid](#section_e86eaf52942c44f5a0aca9f8f68643f2) 545

[SPN for target DC in AD DS](#section_894d09997d794e81a4077bcf6522b0a7) 46

[SPN for target DC in AD LDS](#section_3a6c821d5465414995247bec717fa60a) 47

[SPN target DC in AD DS](#section_41efc56e00074e88bafed7af61efd91f) 44

[SPN target DC in AD LDS](#section_debb73a41e5149f3ac62ae49ce35d13f) 45

[StampLessThanOrEqualUTD](#section_6e7569574acd4f539e9119cfb93475d1) 545

[Standards assignments](#section_063618edb2e24983ab133ed056700641) 42

[StartsWith](#section_6cc630b10d4e4867a851248f4cd0c6db) 545

[State model](#section_0deebc2fa84e40be8a129b3d19b09c45) 49

[String(NT-Sec-Desc)](#section_97b138da550e46ee9c0217936c533852) 449

[String(Sid)](#section_8a8b4cb8e4604611b278c17e0a2a37bb) 449

[String(Teletex)](#section_ebe75cecb5204de18de5cf2c00107a93) 449

[StringSidFromSid](#section_d09e2a96465b4a07926558d92e3e2a19) 545

[SubString](#section_e3a76b8d244b4a0cb4e6847198cace5e) 545

[SVCERR\_DRS\_WIRE\_V1 structure](#section_d05a3b5d494449a993cf95e9f56e20e7) 71

[Syntax](#section_311a3d21e2254fb3bef39a3b10d8c66b) 546

[SYNTAX\_ADDRESS packet](#section_7df24a29d2e44f9eb55cabbd72131422) 546

[SYNTAX\_DISTNAME\_BINARY packet](#section_8eefc5ab6d2248b4bea163b53a81a3a9) 546

[SYSERR\_DRS\_WIRE\_V1 structure](#section_5528c5ea4f694061ae726cea2f7276c3) 71

[systemFlags values](#section_01d618f8003c439cbb64bc19a0f60fc7) 548

T

[Tracking changes](#section_b8dadb6bdc5549e6b1c9afc5e91d20b4) 592

[Transactions](#section_652676b6fac844c4854a963197362ccf) 50

[Transport](#section_e6076eab53f64aad9041888c5734b715) 43

Types ([section 1.3.3](#section_b8c1a431335c4797a5d294569dec581b) 40, [section 3.3.3](#section_a15fa80ed63c4ae29cd9f6df37089c89) 50)

[Typographical conventions](#section_d05f9217c26b4f64b16e2247389319ff) 49

U

[UCHAR](#section_aca7e264341d4231b5ac1003e89e24b7) 548

[ULARGE\_INTEGER structure](#section_686ea1a042d04ff195ef52d0148fe842) 548

[ULONG](#section_20419b45c61d47ccb4fc0b2ab66934cc) 548

[ULONGLONG](#section_4e552c46a7dc4504a902f210e6e6dedd) 548

[UPDERR\_DRS\_WIRE\_V1 structure](#section_c7e91a8c1c264a60b3f64a0d0be368b2) 71

[UPTODATE\_CURSOR\_V1 structure](#section_cf88f341fb494cd5b7e26920cbd91f1b) 549

[UPTODATE\_CURSOR\_V2 structure](#section_d3e30021b6ac413eb08ab69b9b0c6592) 550

[UPTODATE\_VECTOR\_V1\_EXT structure](#section_462b424ab50a4c4aa81f48d0f4cf40fe) 550

[UPTODATE\_VECTOR\_V2\_EXT structure](#section_cebd1ccb891b4268b0564b714cdf981e) 551

[userAccountControl bits](#section_ed5db046d9f74da4884ce14b6bcf5471) 551

[UserNameFromNT4AccountName](#section_82e61ab3fa81432ca64ba18a346a3704) 552

[USHORT](#section_7bbbf218e4ed41d9b0750ed5fdccfed8) 552

[USN](#section_1be1e991a2db4f9199538eab69f60e64) 552

[USN\_VECTOR structure](#section_595d11b86ca74a61bd563e6a2b99b76b) 552

[UUID](#section_fd7db2bb5a0b4a61a0f2dbf0a20ebaa6) 552

V

[Value](#section_c1b732d37bf94ba181ee07157f07294c) 554

[VALUE\_META\_DATA\_EXT\_V1 structure](#section_7530cf2ea2ad4716a5708383f8b1846f) 554

[ValueFromATTRVAL](#section_6d7a4a08a5274e11a559fd5c45dc6c63) 555

Values ([section 5.16.1](#section_e4816252d38c4b5f9821d23bd1dfe296) 445, [section 5.16.2](#section_284c8a5a6ede4d3488babda0b8bb59e0) 447, [section 5.16.3](#section_0d7070d2f71647109f92812dc4cd8a53) 449)

[Variables](#section_c5d9026516534ecca0d7cac691e2d08e) 439

[Vendor-extensible fields](#section_9b545b76247c44aaba77a3581fa4205a) 42

[Versioning](#section_43355596b084491ba3a09176504f2987) 42

W

[WCHAR](#section_b5f107e4d4694263b3d640a40e808e20) 555