**[MS-LSAD]:**

**Local Security Authority (Domain Policy) 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 |
| --- | --- | --- | --- |
| 2/22/2007 | 0.01 | New | Version 0.01 release |
| 6/1/2007 | 1.0 | Major | Updated and revised the technical content. |
| 7/3/2007 | 2.0 | Major | Updated and revised the technical content. |
| 7/20/2007 | 3.0 | Major | Added new content. |
| 8/10/2007 | 4.0 | Major | New content added. |
| 9/28/2007 | 5.0 | Major | Updated and revised the technical content. |
| 10/23/2007 | 5.1 | Minor | Clarified the meaning of the technical content. |
| 11/30/2007 | 5.1.1 | Editorial | Changed language and formatting in the technical content. |
| 1/25/2008 | 6.0 | Major | Updated and revised the technical content. |
| 3/14/2008 | 7.0 | Major | Updated and revised the technical content. |
| 5/16/2008 | 8.0 | Major | Updated and revised the technical content. |
| 6/20/2008 | 9.0 | Major | Updated and revised the technical content. |
| 7/25/2008 | 9.0.1 | Editorial | Changed language and formatting in the technical content. |
| 8/29/2008 | 10.0 | Major | Updated and revised the technical content. |
| 10/24/2008 | 11.0 | Major | Updated and revised the technical content. |
| 12/5/2008 | 12.0 | Major | Updated and revised the technical content. |
| 1/16/2009 | 13.0 | Major | Updated and revised the technical content. |
| 2/27/2009 | 14.0 | Major | Updated and revised the technical content. |
| 4/10/2009 | 15.0 | Major | Updated and revised the technical content. |
| 5/22/2009 | 16.0 | Major | Updated and revised the technical content. |
| 7/2/2009 | 17.0 | Major | Updated and revised the technical content. |
| 8/14/2009 | 18.0 | Major | Updated and revised the technical content. |
| 9/25/2009 | 19.0 | Major | Updated and revised the technical content. |
| 11/6/2009 | 20.0 | Major | Updated and revised the technical content. |
| 12/18/2009 | 21.0 | Major | Updated and revised the technical content. |
| 1/29/2010 | 22.0 | Major | Updated and revised the technical content. |
| 3/12/2010 | 23.0 | Major | Updated and revised the technical content. |
| 4/23/2010 | 23.1 | Minor | Clarified the meaning of the technical content. |
| 6/4/2010 | 24.0 | Major | Updated and revised the technical content. |
| 7/16/2010 | 25.0 | Major | Updated and revised the technical content. |
| 8/27/2010 | 26.0 | Major | Updated and revised the technical content. |
| 10/8/2010 | 27.0 | Major | Updated and revised the technical content. |
| 11/19/2010 | 28.0 | Major | Updated and revised the technical content. |
| 1/7/2011 | 29.0 | Major | Updated and revised the technical content. |
| 2/11/2011 | 30.0 | Major | Updated and revised the technical content. |
| 3/25/2011 | 31.0 | Major | Updated and revised the technical content. |
| 5/6/2011 | 32.0 | Major | Updated and revised the technical content. |
| 6/17/2011 | 33.0 | Major | Updated and revised the technical content. |
| 9/23/2011 | 33.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 12/16/2011 | 34.0 | Major | Updated and revised the technical content. |
| 3/30/2012 | 35.0 | Major | Updated and revised the technical content. |
| 7/12/2012 | 35.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/25/2012 | 36.0 | Major | Updated and revised the technical content. |
| 1/31/2013 | 36.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 8/8/2013 | 37.0 | Major | Updated and revised the technical content. |
| 11/14/2013 | 37.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/13/2014 | 37.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 5/15/2014 | 37.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/30/2015 | 38.0 | Major | Significantly changed the technical content. |
| 10/16/2015 | 38.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/14/2016 | 39.0 | Major | Significantly changed the technical content. |
| 6/1/2017 | 40.0 | Major | Significantly changed the technical content. |
| 9/15/2017 | 41.0 | Major | Significantly changed the technical content. |
| 12/1/2017 | 41.0 | None | No changes to the meaning, language, or formatting of the technical content. |

Table of Contents

[1 Introduction 8](#_Toc499828783)

[1.1 Glossary 8](#_Toc499828784)

[1.2 References 13](#_Toc499828785)

[1.2.1 Normative References 13](#_Toc499828786)

[1.2.2 Informative References 14](#_Toc499828787)

[1.3 Overview 14](#_Toc499828788)

[1.4 Relationship to Other Protocols 19](#_Toc499828789)

[1.5 Prerequisites/Preconditions 21](#_Toc499828790)

[1.6 Applicability Statement 21](#_Toc499828791)

[1.7 Versioning and Capability Negotiation 22](#_Toc499828792)

[1.8 Vendor-Extensible Fields 22](#_Toc499828793)

[1.9 Standards Assignments 22](#_Toc499828794)

[2 Messages 23](#_Toc499828795)

[2.1 Transport 23](#_Toc499828796)

[2.2 Common Data Types 23](#_Toc499828797)

[2.2.1 Constant Value Definitions 26](#_Toc499828798)

[2.2.1.1 ACCESS\_MASK 26](#_Toc499828799)

[2.2.1.1.1 ACCESS\_MASK for All Objects 26](#_Toc499828800)

[2.2.1.1.2 ACCESS\_MASK for Policy Objects 28](#_Toc499828801)

[2.2.1.1.3 ACCESS\_MASK for Account Objects 29](#_Toc499828802)

[2.2.1.1.4 ACCESS\_MASK for Secret Objects 30](#_Toc499828803)

[2.2.1.1.5 ACCESS\_MASK for Trusted Domain Objects 30](#_Toc499828804)

[2.2.1.2 POLICY\_SYSTEM\_ACCESS\_MODE 31](#_Toc499828805)

[2.2.1.3 SECURITY\_INFORMATION 31](#_Toc499828806)

[2.2.2 Basic Data Types 33](#_Toc499828807)

[2.2.2.1 LSAPR\_HANDLE 33](#_Toc499828808)

[2.2.2.2 PLSAPR\_HANDLE 33](#_Toc499828809)

[2.2.2.3 LSA\_UNICODE\_STRING 33](#_Toc499828810)

[2.2.2.4 LSAPR\_OBJECT\_ATTRIBUTES 33](#_Toc499828811)

[2.2.2.5 LSAPR\_SR\_SECURITY\_DESCRIPTOR 34](#_Toc499828812)

[2.2.3 Data Types Referenced by Basic Data Types 34](#_Toc499828813)

[2.2.3.1 STRING 34](#_Toc499828814)

[2.2.3.2 LSAPR\_ACL 35](#_Toc499828815)

[2.2.3.3 SECURITY\_DESCRIPTOR\_CONTROL 35](#_Toc499828816)

[2.2.3.4 LSAPR\_SECURITY\_DESCRIPTOR 35](#_Toc499828817)

[2.2.3.5 SECURITY\_IMPERSONATION\_LEVEL 36](#_Toc499828818)

[2.2.3.6 SECURITY\_CONTEXT\_TRACKING\_MODE 36](#_Toc499828819)

[2.2.3.7 SECURITY\_QUALITY\_OF\_SERVICE 37](#_Toc499828820)

[2.2.4 Policy Query/Set Data Types 37](#_Toc499828821)

[2.2.4.1 POLICY\_INFORMATION\_CLASS 37](#_Toc499828822)

[2.2.4.2 LSAPR\_POLICY\_INFORMATION 38](#_Toc499828823)

[2.2.4.3 POLICY\_AUDIT\_LOG\_INFO 39](#_Toc499828824)

[2.2.4.4 LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO 40](#_Toc499828825)

[2.2.4.5 LSAPR\_POLICY\_PRIMARY\_DOM\_INFO 40](#_Toc499828826)

[2.2.4.6 LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO 41](#_Toc499828827)

[2.2.4.7 LSAPR\_POLICY\_PD\_ACCOUNT\_INFO 41](#_Toc499828828)

[2.2.4.8 POLICY\_LSA\_SERVER\_ROLE 41](#_Toc499828829)

[2.2.4.9 POLICY\_LSA\_SERVER\_ROLE\_INFO 42](#_Toc499828830)

[2.2.4.10 LSAPR\_POLICY\_REPLICA\_SRCE\_INFO 42](#_Toc499828831)

[2.2.4.11 POLICY\_MODIFICATION\_INFO 42](#_Toc499828832)

[2.2.4.12 POLICY\_AUDIT\_FULL\_SET\_INFO 42](#_Toc499828833)

[2.2.4.13 POLICY\_AUDIT\_FULL\_QUERY\_INFO 43](#_Toc499828834)

[2.2.4.14 LSAPR\_POLICY\_DNS\_DOMAIN\_INFO 43](#_Toc499828835)

[2.2.4.15 POLICY\_DOMAIN\_INFORMATION\_CLASS 43](#_Toc499828836)

[2.2.4.16 LSAPR\_POLICY\_DOMAIN\_INFORMATION 44](#_Toc499828837)

[2.2.4.17 POLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO 44](#_Toc499828838)

[2.2.4.18 LSAPR\_POLICY\_DOMAIN\_EFS\_INFO 44](#_Toc499828839)

[2.2.4.19 POLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO 45](#_Toc499828840)

[2.2.4.20 POLICY\_AUDIT\_EVENT\_TYPE 46](#_Toc499828841)

[2.2.5 Account Query/Set Data Types 46](#_Toc499828842)

[2.2.5.1 LSAPR\_ACCOUNT\_INFORMATION 46](#_Toc499828843)

[2.2.5.2 LSAPR\_ACCOUNT\_ENUM\_BUFFER 46](#_Toc499828844)

[2.2.5.3 LSAPR\_USER\_RIGHT\_SET 47](#_Toc499828845)

[2.2.5.4 LSAPR\_LUID\_AND\_ATTRIBUTES 47](#_Toc499828846)

[2.2.5.5 LSAPR\_PRIVILEGE\_SET 47](#_Toc499828847)

[2.2.6 Secret Query/Set Data Types 48](#_Toc499828848)

[2.2.6.1 LSAPR\_CR\_CIPHER\_VALUE 48](#_Toc499828849)

[2.2.7 Trusted Domain Query/Set Data Types 48](#_Toc499828850)

[2.2.7.1 LSAPR\_TRUST\_INFORMATION 48](#_Toc499828851)

[2.2.7.2 TRUSTED\_INFORMATION\_CLASS 49](#_Toc499828852)

[2.2.7.3 LSAPR\_TRUSTED\_DOMAIN\_INFO 50](#_Toc499828853)

[2.2.7.4 LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO 51](#_Toc499828854)

[2.2.7.5 LSAPR\_TRUSTED\_CONTROLLERS\_INFO 51](#_Toc499828855)

[2.2.7.6 TRUSTED\_POSIX\_OFFSET\_INFO 52](#_Toc499828856)

[2.2.7.7 LSAPR\_TRUSTED\_PASSWORD\_INFO 52](#_Toc499828857)

[2.2.7.8 LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_BASIC 52](#_Toc499828858)

[2.2.7.9 LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX 52](#_Toc499828859)

[2.2.7.10 LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 54](#_Toc499828860)

[2.2.7.11 LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION 55](#_Toc499828861)

[2.2.7.12 LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL 56](#_Toc499828862)

[2.2.7.13 LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION 56](#_Toc499828863)

[2.2.7.14 LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL 56](#_Toc499828864)

[2.2.7.15 LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2 57](#_Toc499828865)

[2.2.7.16 LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB 57](#_Toc499828866)

[2.2.7.17 LSAPR\_AUTH\_INFORMATION 59](#_Toc499828867)

[2.2.7.18 TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES 60](#_Toc499828868)

[2.2.7.19 LSAPR\_TRUSTED\_ENUM\_BUFFER 61](#_Toc499828869)

[2.2.7.20 LSAPR\_TRUSTED\_ENUM\_BUFFER\_EX 61](#_Toc499828870)

[2.2.7.21 LSA\_FOREST\_TRUST\_RECORD 61](#_Toc499828871)

[2.2.7.22 LSA\_FOREST\_TRUST\_RECORD\_TYPE 63](#_Toc499828872)

[2.2.7.23 LSA\_FOREST\_TRUST\_BINARY\_DATA 63](#_Toc499828873)

[2.2.7.24 LSA\_FOREST\_TRUST\_DOMAIN\_INFO 63](#_Toc499828874)

[2.2.7.25 LSA\_FOREST\_TRUST\_INFORMATION 64](#_Toc499828875)

[2.2.7.26 LSA\_FOREST\_TRUST\_COLLISION\_RECORD\_TYPE 64](#_Toc499828876)

[2.2.7.27 LSA\_FOREST\_TRUST\_COLLISION\_RECORD 64](#_Toc499828877)

[2.2.7.28 LSA\_FOREST\_TRUST\_COLLISION\_INFORMATION 65](#_Toc499828878)

[2.2.8 Privilege Data Types 65](#_Toc499828879)

[2.2.8.1 LSAPR\_POLICY\_PRIVILEGE\_DEF 65](#_Toc499828880)

[2.2.8.2 LSAPR\_PRIVILEGE\_ENUM\_BUFFER 65](#_Toc499828881)

[2.3 Directory Service Schema Elements 66](#_Toc499828882)

[3 Protocol Details 67](#_Toc499828883)

[3.1 Server Details 67](#_Toc499828884)

[3.1.1 Abstract Data Model 67](#_Toc499828885)

[3.1.1.1 Policy Object Data Model 67](#_Toc499828886)

[3.1.1.2 Accounts Rights Data Model 69](#_Toc499828887)

[3.1.1.2.1 Privilege Data Model 69](#_Toc499828888)

[3.1.1.2.2 System Access Rights Data Model 72](#_Toc499828889)

[3.1.1.3 Account Object Data Model 73](#_Toc499828890)

[3.1.1.4 Secret Object Data Model 73](#_Toc499828891)

[3.1.1.5 Trusted Domain Object Data Model 75](#_Toc499828892)

[3.1.1.6 Configuration Settings 76](#_Toc499828893)

[3.1.1.6.1 Block Anonymous Access to Objects 76](#_Toc499828894)

[3.1.1.7 LsaContextHandle Data Model 76](#_Toc499828895)

[3.1.1.8 Attribute Listing 77](#_Toc499828896)

[3.1.1.9 Object Class Listing 77](#_Toc499828897)

[3.1.1.10 Access for Public Abstract Data Model Elements 78](#_Toc499828898)

[3.1.1.10.1 Example Patterns for Direct Access of Policy Object ADM Elements 78](#_Toc499828899)

[3.1.1.10.1.1 Query Pattern for Policy Object ADM 78](#_Toc499828900)

[3.1.1.10.1.2 Set Pattern for Policy Object ADM 79](#_Toc499828901)

[3.1.2 Timers 79](#_Toc499828902)

[3.1.3 Initialization 79](#_Toc499828903)

[3.1.4 Message Processing Events and Sequencing Rules 79](#_Toc499828904)

[3.1.4.1 Obtaining Handles 84](#_Toc499828905)

[3.1.4.2 Access Rights and Access Checks 84](#_Toc499828906)

[3.1.4.2.1 Access Checks Applied on Handle Open 86](#_Toc499828907)

[3.1.4.2.2 Access Checks Applied for Object Operations 87](#_Toc499828908)

[3.1.4.2.3 Determining If Requestors Are Anonymous 87](#_Toc499828909)

[3.1.4.3 Closing Handles 88](#_Toc499828910)

[3.1.4.4 Policy Object Methods 88](#_Toc499828911)

[3.1.4.4.1 LsarOpenPolicy2 (Opnum 44) 88](#_Toc499828912)

[3.1.4.4.2 LsarOpenPolicy (Opnum 6) 89](#_Toc499828913)

[3.1.4.4.3 LsarQueryInformationPolicy2 (Opnum 46) 90](#_Toc499828914)

[3.1.4.4.4 LsarQueryInformationPolicy (Opnum 7) 92](#_Toc499828915)

[3.1.4.4.5 LsarSetInformationPolicy2 (Opnum 47) 93](#_Toc499828916)

[3.1.4.4.6 LsarSetInformationPolicy (Opnum 8) 95](#_Toc499828917)

[3.1.4.4.7 LsarQueryDomainInformationPolicy (Opnum 53) 96](#_Toc499828918)

[3.1.4.4.8 LsarSetDomainInformationPolicy (Opnum 54) 97](#_Toc499828919)

[3.1.4.5 Account Object Methods 98](#_Toc499828920)

[3.1.4.5.1 LsarCreateAccount (Opnum 10) 99](#_Toc499828921)

[3.1.4.5.2 LsarEnumerateAccounts (Opnum 11) 100](#_Toc499828922)

[3.1.4.5.3 LsarOpenAccount (Opnum 17) 102](#_Toc499828923)

[3.1.4.5.4 LsarEnumeratePrivilegesAccount (Opnum 18) 103](#_Toc499828924)

[3.1.4.5.5 LsarAddPrivilegesToAccount (Opnum 19) 104](#_Toc499828925)

[3.1.4.5.6 LsarRemovePrivilegesFromAccount (Opnum 20) 105](#_Toc499828926)

[3.1.4.5.7 LsarGetSystemAccessAccount (Opnum 23) 105](#_Toc499828927)

[3.1.4.5.8 LsarSetSystemAccessAccount (Opnum 24) 106](#_Toc499828928)

[3.1.4.5.9 LsarEnumerateAccountsWithUserRight (Opnum 35) 107](#_Toc499828929)

[3.1.4.5.10 LsarEnumerateAccountRights (Opnum 36) 108](#_Toc499828930)

[3.1.4.5.11 LsarAddAccountRights (Opnum 37) 109](#_Toc499828931)

[3.1.4.5.12 LsarRemoveAccountRights (Opnum 38) 110](#_Toc499828932)

[3.1.4.6 Secret Object Methods 111](#_Toc499828933)

[3.1.4.6.1 LsarCreateSecret (Opnum 16) 112](#_Toc499828934)

[3.1.4.6.2 LsarOpenSecret (Opnum 28) 113](#_Toc499828935)

[3.1.4.6.3 LsarSetSecret (Opnum 29) 115](#_Toc499828936)

[3.1.4.6.4 LsarQuerySecret (Opnum 30) 116](#_Toc499828937)

[3.1.4.6.5 LsarStorePrivateData (Opnum 42) 117](#_Toc499828938)

[3.1.4.6.6 LsarRetrievePrivateData (Opnum 43) 118](#_Toc499828939)

[3.1.4.7 Trusted Domain Object Methods 119](#_Toc499828940)

[3.1.4.7.1 LsarOpenTrustedDomain (Opnum 25) 120](#_Toc499828941)

[3.1.4.7.2 LsarQueryTrustedDomainInfo (Opnum 39) 121](#_Toc499828942)

[3.1.4.7.3 LsarSetTrustedDomainInfo (Opnum 40) 122](#_Toc499828943)

[3.1.4.7.4 LsarDeleteTrustedDomain (Opnum 41) 125](#_Toc499828944)

[3.1.4.7.5 LsarQueryTrustedDomainInfoByName (Opnum 48) 126](#_Toc499828945)

[3.1.4.7.6 LsarSetTrustedDomainInfoByName (Opnum 49) 127](#_Toc499828946)

[3.1.4.7.7 LsarEnumerateTrustedDomainsEx (Opnum 50) 128](#_Toc499828947)

[3.1.4.7.8 LsarEnumerateTrustedDomains (Opnum 13) 129](#_Toc499828948)

[3.1.4.7.9 LsarOpenTrustedDomainByName (Opnum 55) 131](#_Toc499828949)

[3.1.4.7.10 LsarCreateTrustedDomainEx2 (Opnum 59) 132](#_Toc499828950)

[3.1.4.7.11 LsarCreateTrustedDomainEx (Opnum 51) 134](#_Toc499828951)

[3.1.4.7.12 LsarCreateTrustedDomain (Opnum 12) 136](#_Toc499828952)

[3.1.4.7.13 LsarQueryInfoTrustedDomain (Opnum 26) 137](#_Toc499828953)

[3.1.4.7.14 LsarSetInformationTrustedDomain (Opnum 27) 139](#_Toc499828954)

[3.1.4.7.15 LsarQueryForestTrustInformation (Opnum 73) 142](#_Toc499828955)

[3.1.4.7.16 LsarSetForestTrustInformation (Opnum 74) 143](#_Toc499828956)

[3.1.4.7.16.1 Forest Trust Collision Generation 145](#_Toc499828957)

[3.1.4.8 Privilege Methods 146](#_Toc499828958)

[3.1.4.8.1 LsarEnumeratePrivileges (Opnum 2) 147](#_Toc499828959)

[3.1.4.8.2 LsarLookupPrivilegeValue (Opnum 31) 148](#_Toc499828960)

[3.1.4.8.3 LsarLookupPrivilegeName (Opnum 32) 149](#_Toc499828961)

[3.1.4.8.4 LsarLookupPrivilegeDisplayName (Opnum 33) 150](#_Toc499828962)

[3.1.4.9 Common Object Methods 151](#_Toc499828963)

[3.1.4.9.1 LsarQuerySecurityObject (Opnum 3) 151](#_Toc499828964)

[3.1.4.9.2 LsarSetSecurityObject (Opnum 4) 153](#_Toc499828965)

[3.1.4.9.3 LsarDeleteObject (Opnum 34) 154](#_Toc499828966)

[3.1.4.9.4 LsarClose (Opnum 0) 155](#_Toc499828967)

[3.1.4.10 Data Validation 156](#_Toc499828968)

[3.1.5 Timer Events 160](#_Toc499828969)

[3.1.6 Other Local Events 160](#_Toc499828970)

[3.1.6.1 LSAPR\_HANDLE\_rundown 160](#_Toc499828971)

[4 Protocol Examples 161](#_Toc499828972)

[4.1 Manipulating Account Objects 161](#_Toc499828973)

[4.2 Manipulating Secret Objects 164](#_Toc499828974)

[4.3 Manipulating Trusted Domain Objects 167](#_Toc499828975)

[4.4 Structure Example of LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB 169](#_Toc499828976)

[5 Security 173](#_Toc499828977)

[5.1 Security Considerations for Implementers 173](#_Toc499828978)

[5.1.1 RC4 Cipher Usage 173](#_Toc499828979)

[5.1.2 Secret Encryption and Decryption 173](#_Toc499828980)

[5.1.3 DES-ECB-LM Cipher Definition 175](#_Toc499828981)

[5.1.4 Encryption and Decryption Examples 175](#_Toc499828982)

[5.1.4.1 Encryption Example 176](#_Toc499828983)

[5.1.4.2 Decryption Example 176](#_Toc499828984)

[5.2 Index of Security Parameters 177](#_Toc499828985)

[6 Appendix A: Full IDL 178](#_Toc499828986)

[7 Appendix B: Product Behavior 194](#_Toc499828987)

[8 Change Tracking 216](#_Toc499828988)

[9 Index 217](#_Toc499828989)

# Introduction

The Local Security Authority (Domain Policy) Remote Protocol is used to manage various machine and [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) security policies. All versions of Windows NT operating system–based products, in all configurations, implement and listen on the server side of this protocol. However, not all operations are meaningful in all configurations.

This protocol, with minor exceptions, enables remote policy-management scenarios. Therefore, the majority of this interface does not need to be implemented to achieve Windows client-to-server ([**domain controller**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) configuration and otherwise) interoperability, as defined by the ability for Windows clients to retrieve policy settings from servers.

Policy settings controlled by this protocol relate to the following:

* **Account objects**: The rights and [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) that [**security principals**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) have on the server.
* **Secret objects**: Mechanisms that securely store data on the server.
* **Trusted domain objects**: Mechanisms that the Windows operating system uses for describing [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) relationships between domains and [**forests**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62).
* Other miscellaneous settings, such as lifetimes of Kerberos tickets, states of domain controller (backup or primary), and other unrelated pieces of policy.

All of these types of policy are addressed in sections of this document that specify the server data model.

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

## Glossary

This document uses the following terms:

**64-bit Network Data Representation (NDR64)**: A specific instance of a remote procedure call (RPC) transfer syntax. For more information about RPC transfer syntax, see [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 14.

**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.

**account domain**: A [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), identified by a [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d), that is the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) namespace for which a given machine is authoritative. The [**account domain**](#gt_b56f14e3-d874-48bc-837b-5e812ee1a96e) is the same as the [**primary domain**](#gt_387021de-3d6b-4372-835f-0d68c50cb496) for a [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) and is its default [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). For a machine that is joined to a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), the [**account domain**](#gt_b56f14e3-d874-48bc-837b-5e812ee1a96e) is the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) namespace defined by the local Security Accounts Manager [[MS-SAMR]](%5bMS-SAMR%5d.pdf#Section_4df07fab1bbc452f8e927853a3c7e380).

**account object**: An element of a Local Security Authority (LSA) policy database that describes the rights and privileges granted by the server to a [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409). The [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) matches that of the [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b).

**ACID**: A term that refers to the four properties that any database system must achieve in order to be considered transactional: Atomicity, Consistency, Isolation, and Durability [GRAY].

**Active Directory**: A general-purpose network [**directory service**](#gt_c36db657-3138-4d9a-9289-ded5cbb8b40e). [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) also refers to the Windows implementation of a [**directory service**](#gt_c36db657-3138-4d9a-9289-ded5cbb8b40e). [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) stores information about a variety of objects 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) or Active Directory Lightweight Directory Services (AD LDS). [[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) versions 2 and 3, Kerberos, and DNS.

**backup domain controller (BDC)**: A [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that receives a copy of the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) directory database from the [**primary domain controller (PDC)**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d). This copy is synchronized periodically and automatically with the [**primary domain controller (PDC)**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d). BDCs also authenticate user logons and can be promoted to function as the [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d). There is only one [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) or [**PDC**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d) emulator in a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), and the rest are [**backup domain controllers**](#gt_ce1138c6-7ab4-4c37-98b4-95599071c3c3).

**Coordinated Universal Time (UTC)**: A high-precision atomic time standard that approximately tracks Universal Time (UT). It is the basis for legal, civil time all over the Earth. Time zones around the world are expressed as positive and negative offsets from UTC. In this role, it is also referred to as Zulu time (Z) and Greenwich Mean Time (GMT). In these specifications, all references to UTC refer to the time at UTC-0 (or GMT).

**directory**: The database that stores information about objects such as users, groups, computers, printers, and the [**directory service**](#gt_c36db657-3138-4d9a-9289-ded5cbb8b40e) that makes this information available to users and applications.

**directory service (DS)**: A service that stores and organizes information about a computer network's users and network shares, and that allows network administrators to manage users' access to the shares. See also [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90).

**discretionary access control list (DACL)**: An [**access control list (ACL)**](#gt_9f92aa05-dd0a-45f2-88d6-89f1fb654395) that is controlled by the owner of an object and that specifies the access particular users or groups can have to the object.

**DNS name**: A fully qualified domain name (FQDN).

**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 and interoperates with other [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) to ensure that a local change to an object 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), the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) contains full NC replicas of the configuration naming context (config NC), schema naming context (schema NC), 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 [**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 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), several AD LDS [**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, only one AD DS [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) can run on one server. However, several AD LDS [**DCs**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) can coexist with one AD DS [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) on one server. The AD LDS [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) contains full NC replicas of the config NC and the schema NC 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 member (member machine)**: A machine that is joined to a domain by sharing a secret between the machine and the domain.

**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), 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 can contain security principal objects. [**Domain NCs**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) appear in the global catalog (GC). 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. In AD DS, 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 domainDNS; for directory replication [[MS-DRSR]](%5bMS-DRSR%5d.pdf#Section_f977faaa673e4f66b9bf48c640241d47), see domainDNS.

**endpoint**: A network-specific address of a [**remote procedure call (RPC) server**](#gt_ae65dac0-cd24-4e83-a946-6d1097b71553) 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].

**forest**: One or more [**domains**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) that share a common schema and trust each other transitively. An organization can have multiple [**forests**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). A [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) establishes the security and administrative boundary for all the objects that reside within the [**domains**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) that belong to the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). In contrast, a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) establishes the administrative boundary for managing objects, such as users, groups, and computers. In addition, each [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) has individual security policies and trust relationships with other [**domains**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

**forest functional level**: A specification of functionality available in a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). It must be less than or equal to the [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) functional level of every [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) in the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). See [MS-ADTS] section 6.1.4.4 for information on how the [**forest functional level**](#gt_b3240417-ca43-4901-90ec-fde55b32b3b8) is determined.

**forest trust**: A type of [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) where the trusted party is a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62), which means that all [**domains**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) in that [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) are trusted.

**forest trust information**: Information about namespaces, [**domain names**](#gt_45a1c9f1-0263-49a8-97c7-7aca1a99308c), and [**security identifiers (SIDs)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) owned by a trusted [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62).

**global catalog server (GC server)**: A [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) that contains a naming context (NC) replica (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).

**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).

**interdomain trust account**: An account that stores information associated with a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) in the [**domain controllers (DCs)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) of the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) that is trusted to perform authentication.

**local account domain**: A [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), identified by a [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d), that is a [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) namespace for which a given machine is authoritative. The [**local account domain**](#gt_5127d055-89b1-49ba-adf0-70470d9b9da0) is the same as the [**account domain**](#gt_b56f14e3-d874-48bc-837b-5e812ee1a96e) for any non–[**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd). On a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), the [**local account domain**](#gt_5127d055-89b1-49ba-adf0-70470d9b9da0) is an [**account domain**](#gt_b56f14e3-d874-48bc-837b-5e812ee1a96e) local to the [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

**locally unique identifier (LUID)**: A 64-bit value guaranteed to be unique within the scope of a single machine.

**Network Data Representation (NDR)**: A specification that defines a mapping from Interface Definition Language (IDL) data types onto octet streams. [**NDR**](#gt_9ebf9540-2c31-43bc-bc56-4a932faabf2d) also refers to the runtime environment that implements the mapping facilities (for example, data provided to [**NDR**](#gt_9ebf9540-2c31-43bc-bc56-4a932faabf2d)). For more information, see [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) and [C706] section 14.

**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].

**original equipment manufacturer (OEM) code page**: A code page used to translate between non-Unicode encoded strings and UTF-16 encoded strings.

**primary domain**: A [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) (identified by a [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d)) that the server is joined to. For a [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), the [**primary domain**](#gt_387021de-3d6b-4372-835f-0d68c50cb496) is that of the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) itself.

**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).

**privilege**: The capability of a security principal to perform a type of operation on a computer system regardless of restrictions placed by discretionary access control.

**RC4**: A variable key-length symmetric encryption algorithm. For more information, see [[SCHNEIER]](https://go.microsoft.com/fwlink/?LinkId=817338) section 17.1.

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

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

**root domain**: The unique [**domain naming contexts (domain NCs)**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) of an Active Directory [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) that is the parent of the [**forest's**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) config NC. The config NC's relative distinguished name (RDN) is "cn=Configuration" relative to the root object of the root domain. The root domain is the domain that is created first in a [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62).

**RPC client**: A computer on the network that sends messages using remote procedure call (RPC) as its transport, waits for responses, and is the initiator in an RPC exchange.

**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 server**: A computer on the network that waits for messages, processes them when they arrive, and sends responses using RPC as its transport acts as the responder during a remote procedure call (RPC) exchange.

**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.

**secret object**: An element of the Local Security Authority (LSA) Policy Database, which contains a value that is secret in that access to it is strictly controlled through cryptographic protections and restrictive access control mechanisms.

**security descriptor**: A data structure containing the security information associated with a securable object. A [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) identifies an object's owner by its [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d). If access control is configured for the object, its [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) contains a [**discretionary access control list (DACL)**](#gt_d727f612-7a45-48e4-9d87-71735d62b321) 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 security status. The [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) is used to guard access to an object as well as to control which type of auditing takes place when the object 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). 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).

**Server Message Block (SMB)**: A protocol that is used to request file and print services from server systems over a network. The SMB protocol extends the CIFS protocol with additional security, file, and disk management support. For more information, see [[CIFS]](https://go.microsoft.com/fwlink/?LinkId=89836) and [[MS-SMB]](%5bMS-SMB%5d.pdf#Section_f210069c70864dc2885e861d837df688).

**server role**: The state of a [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), which can be one of two values--primary DC or backup DC.

**service**: A process or agent that is available on the network, offering resources or services for clients. Examples of services include file servers, web servers, and so on.

**system access control list (SACL)**: An [**access control list (ACL)**](#gt_9f92aa05-dd0a-45f2-88d6-89f1fb654395) that controls the generation of audit messages for attempts to access a securable object. The ability to get or set an object's [**SACL**](#gt_c189801e-3752-4715-88f4-17804dad5782) is controlled by a privilege typically held only by system administrators.

**trust**: To accept another authority's statements for the purposes of authentication and authorization, especially in the case of a relationship between two domains. If [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) A trusts [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) B, [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) A accepts [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) B's authentication and authorization statements for principals represented by security principal objects in [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) B; for example, the list of groups to which a particular user belongs. As a noun, a [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) is the relationship between two [**domains**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) described in the previous sentence.

**trust attributes**: A collection of attributes that define different characteristics of a trust within a domain or a forest.

**trusted domain**: A domain that is trusted to make authentication decisions for security principals in that domain.

**trusted domain object (TDO)**: A collection of properties that define a trust relationship with another domain, such as direction (outbound, inbound, or both), trust attributes, name, and security identifier of the other domain. For more information, see [MS-ADTS].

**trusted forest**: A forest that is trusted to make authentication statements for security principals in that forest. Assuming forest A trusts forest B, all domains belonging to forest A will trust all domains in forest B, subject to policy configuration.

**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.

**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)

[FIPS81] FIPS PUBS, "DES Modes of Operation", December 1980, [http://csrc.nist.gov/publications/fips/fips81/fips81.htm](https://go.microsoft.com/fwlink/?LinkId=89874)

[GRAY] Gray, J., and Reuter, A., "Transaction Processing: Concepts and Techniques", The Morgan Kaufmann Series in Data Management Systems, San Francisco: Morgan Kaufmann Publishers, 1992, Hardcover ISBN: 9781558601901.

[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-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-GPEF] Microsoft Corporation, "[Group Policy: Encrypting File System Extension](%5bMS-GPEF%5d.pdf#Section_14d3fd83753741a2af398e52c19ef0e3)".

[MS-GPSB] Microsoft Corporation, "[Group Policy: Security Protocol Extension](%5bMS-GPSB%5d.pdf#Section_6a07a06be62847659d910d63ba47fdc0)".

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

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

[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-SMB2] Microsoft Corporation, "[Server Message Block (SMB) Protocol Versions 2 and 3](%5bMS-SMB2%5d.pdf#Section_5606ad475ee0437a817e70c366052962)".

[MS-SMB] Microsoft Corporation, "[Server Message Block (SMB) Protocol](%5bMS-SMB%5d.pdf#Section_f210069c70864dc2885e861d837df688)".

[MS-WKST] Microsoft Corporation, "[Workstation Service Remote Protocol](%5bMS-WKST%5d.pdf#Section_5bb08058bc364d3cabebb132228281b7)".

[MSKB-3149090] Microsoft Corporation, "MS16-047: Description of the security update for SAM and LSAD remote protocols", April 2016, [https://support.microsoft.com/en-us/kb/3149090](https://go.microsoft.com/fwlink/?LinkId=786963)

[MSKB-3155495] Microsoft Corporation, "You can't use the Active Directory shadow principal groups feature for groups that are always filtered out in Windows", revision 2.0, May 2016, [https://support.microsoft.com/en-us/kb/3155495](https://go.microsoft.com/fwlink/?LinkId=808755)

[RFC1088] McLaughlin III, L., "A Standard for the Transmission of IP Datagrams over NetBIOS Networks", RFC 1088, February 1989, [http://www.ietf.org/rfc/rfc1088.txt](https://go.microsoft.com/fwlink/?LinkId=90266)

[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)

[RFC3961] Raeburn, K., "Encryption and Checksum Specifications for Kerberos 5", RFC 3961, February 2005, [http://www.ietf.org/rfc/rfc3961.txt](https://go.microsoft.com/fwlink/?LinkId=90450)

[RFC4120] Neuman, C., Yu, T., Hartman, S., and Raeburn, K., "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005, [http://www.rfc-editor.org/rfc/rfc4120.txt](https://go.microsoft.com/fwlink/?LinkId=90458)

[RFC4757] Jaganathan, K., Zhu, L., and Brezak, J., "The RC4-HMAC Kerberos Encryption Types Used by Microsoft Windows", RFC 4757, December 2006, [http://www.ietf.org/rfc/rfc4757.txt](https://go.microsoft.com/fwlink/?LinkId=90488)

### Informative References

[MS-DRSR] Microsoft Corporation, "[Directory Replication Service (DRS) Remote Protocol](%5bMS-DRSR%5d.pdf#Section_f977faaa673e4f66b9bf48c640241d47)".

[MSFT-LSA-IDL] Microsoft Corporation, "Local Security Authority Merged IDL File", December 2015, [http://www.microsoft.com/downloads/details.aspx?displaylang=en&familyID=7700ad04-866b-447a-9e08-21dbda94460f](https://go.microsoft.com/fwlink/?LinkId=150268)

[SCHNEIER] Schneier, B., "Applied Cryptography, Second Edition", John Wiley and Sons, 1996, ISBN: 0471117099, [http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471117099.html](https://go.microsoft.com/fwlink/?LinkId=817338)

## Overview

The Local Security Authority (Domain Policy) Remote Protocol provides a [**remote procedure call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface used for providing remote management for policy settings related to [**account objects**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b), [**secret objects**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d), [**trusted domain objects (TDOs)**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4), and other miscellaneous security-related policy settings. The client end of the Local Security Authority (Domain Policy) Remote Protocol is an application that issues method calls on the RPC interface. The server end of the Local Security Authority (Domain Policy) Remote Protocol is a [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) that implements support for this RPC interface.

The following represent primary use cases for remote management:

* Creating, deleting, enumerating, and modifying [**trusts**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6), account objects, and secret objects.
* Querying and modifying policy settings unrelated to TDOs, account objects or secret objects, such as lifetimes of Kerberos tickets.

This protocol is used by Windows clients for the "domain join" operation (as specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.4) as an implementation choice to achieve the end state, as specified in [MS-ADTS]. The specific profile of the Local Security Authority (Domain Policy) Remote Protocol for the "domain join" scenario is specified in section [1.6](#Section_7c9812deb31d47a1a13fa91ae94fad37) as "Retrieval of policy settings by clients".

The server end of the Local Security Authority (Domain Policy) Remote Protocol can be implemented on a [**domain controller (DC)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), including [**primary domain controllers (PDCs)**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d), [**backup domain controllers (BDCs)**](#gt_ce1138c6-7ab4-4c37-98b4-95599071c3c3), [**global catalog servers (GC servers)**](#gt_a5a99ce4-e206-42dc-8874-e103934c5b0d), and [**read-only domain controllers (RODCs)**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870), or on a non–domain controller. In the case of a DC, including PDCs, BDCs, GC servers, and RODCs, the server end of this protocol can be in one of the [**forest functional levels**](#gt_b3240417-ca43-4901-90ec-fde55b32b3b8). The behavior of the server end of the Local Security Authority (Domain Policy) Remote Protocol is the same in these cases, except when noted in the message processing descriptions for the methods of this protocol. See sections [3.1.4.4.1](#Section_9456a9637c214710af77d0a2f5a72d6b), [3.1.4.4.3](#Section_516f503c0230489db012e650b46b66a2), [3.1.4.4.5](#Section_fc14e9aea26e4031809ea908dd3e13a3), [3.1.4.7](#Section_3d7af62ec260489d8ab5c756e0b9fe57), [3.1.4.7.3](#Section_d541b40a06ad4b3bbbfa5cf51a1f02d9), [3.1.4.7.4](#Section_633788c91e984555bc41d78a0de0f4a4), [3.1.4.7.10](#Section_cc86a55db61948fd998a65cca15efeb9), [3.1.4.7.14](#Section_9ea46cefcc724109ba1391eda6b713bc), and [3.1.4.7.16](#Section_16be42bce85c4135b183aacb88106306) for details.

This protocol is a simple request/response-based RPC protocol. Typically, there are no long-lived sessions, although clients can cache the RPC connection and reuse it over time. A sample sequence of requests and responses is specified in section [4](#Section_3186df57d57c4dc380a77f3050a5ccca).

It is helpful to consider two perspectives when understanding and implementing this protocol: an object-based perspective and a method-based perspective.

The object-based perspective shows that the protocol exposes four main object abstractions: a policy object, an account object, a secret object, and a trusted domain object. A requester obtains a "handle" (an RPC context handle) to one of these objects and then performs one or more actions on the object. The following is a brief listing of methods that operate on each of the respective object types.

Policy object:

* LsarOpenPolicy2
* LsarQueryInformationPolicy2
* LsarSetInformationPolicy2
* LsarClose
* LsarQueryDomainInformationPolicy
* LsarEnumeratePrivileges
* LsarLookupPrivilegeName
* LsarLookupPrivilegeValue
* LsarLookupPrivilegeDisplayName
* LsarSetDomainInformationPolicy
* LsarQuerySecurityObject
* LsarSetSecurityObject

Account object:

* LsarCreateAccount
* LsarOpenAccount
* LsarEnumerateAccounts
* LsarClose
* LsarDeleteObject
* LsarSetSystemAccessAccount
* LsarQuerySecurityObject
* LsarAddAccountRights
* LsarRemoveAccountRights
* LsarAddPrivilegesToAccount
* LsarRemovePrivilegesFromAccount
* LsarEnumerateAccountsWithUserRight
* LsarGetSystemAccessAccount
* LsarSetSecurityObject
* LsarEnumeratePrivilegesAccount
* LsarEnumerateAccountRights

Secret object:

* LsarCreateSecret
* LsarOpenSecret
* LsarClose
* LsarDeleteObject
* LsarRetrievePrivateData
* LsarStorePrivateData
* LsarSetSecret
* LsarQuerySecret
* LsarQuerySecurityObject
* LsarSetSecurityObject

Trusted domain object:

* LsarCreateTrustedDomainEx2
* LsarOpenTrustedDomain
* LsarClose
* LsarDeleteObject
* LsarOpenTrustedDomainByName
* LsarDeleteTrustedDomain
* LsarEnumerateTrustedDomainsEx
* LsarQueryInfoTrustedDomain
* LsarSetInformationTrustedDomain
* LsarQueryForestTrustInformation
* LsarSetForestTrustInformation
* LsarQueryTrustedDomainInfo
* LsarSetTrustedDomainInfo
* LsarQueryTrustedDomainInfoByName
* LsarSetTrustedDomainInfoByName

For example, to set a policy that controls the lifetime of Kerberos tickets, a requester opens a handle to the Policy object and updates the maximum service ticket age policy setting via a parameter called *MaxServiceTicketAge*. The call sequence from the requester appears as follows (with the parameter information removed for brevity):

1. Send LsarOpenPolicy2 request; receive LsarOpenPolicy2 reply.
2. Send LsarQueryDomainInformationPolicy request; receive LsarQueryDomainInformationPolicy reply.
3. Send LsarSetDomainInformationPolicy request; receive LsarSetDomainInformationPolicy reply.
4. Send LsarClose request; receive LsarClose reply.

The following is a brief explanation of the call sequence:

1. Using the network address of a responder that implements this protocol, a requester makes an LsarOpenPolicy2 request to obtain a handle to the policy object. This handle is necessary to examine and manipulate [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) policy information.
2. Using the handle returned from LsarOpenPolicy2, the requester makes an LsarQueryDomainInformationPolicy request to retrieve the current policy settings that affect Kerberos tickets.
3. After modifying the portions of the Kerberos ticket policy information to suit the requester, the requester makes the LsarSetDomainInformationPolicy request to set the policy to the new values.
4. The requester closes the policy handle returned from LsarOpenPolicy2. This releases responder resources associated with the handle.

In the method-based perspective, there is a common set of operations for each object type. The operations fall into patterns. The following is a list of the patterns and associated methods, along with a description of the pattern.

* **Open pattern**: This pattern returns an RPC context handle that references a specific object type. A requester uses this pattern by specifying a specific access for the handle in the request and using the returned handle to call other methods that require the returned handle and the associated access. For example, calling the LsarSetSecret method requires a secret object handle that has been opened with SECRET\_WRITE access.

LsarOpenPolicy2 is distinguished from the other methods in this pattern in two ways. First, the requestor calls this method before calling any other handle-based methods. Second, a network address, rather than a context handle, is required to indicate the responder.

The following are the methods that follow the open pattern:

* + LsarOpenPolicy2
  + LsarOpenPolicy
  + LsarOpenAccount
  + LsarOpenSecret
  + LsarOpenTrustedDomain
  + LsarOpenTrustedDomainByName
* **Enumerate pattern**: This pattern enables a requester to obtain a complete listing of all objects of a certain type (account or [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4)) or to get all values of a certain type out of an object (for example, [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) known to the server).

The following are the methods that follow the enumerate pattern:

* + LsarEnumerateTrustedDomainsEx
  + LsarEnumerateAccounts
  + LsarEnumeratePrivileges
  + LsarEnumeratePrivilegesAccount
  + LsarEnumerateAccountRights
  + LsarEnumerateAccountsWithUserRight
* **Create pattern**: Methods in this pattern enable specified objects to be created. A handle to the newly created object is also returned.

The following are the methods that follow the create pattern:

* + LsarCreateAccount
  + LsarCreateSecret
  + LsarCreateTrustedDomainEx2
* **Query pattern**: This pattern enables specified attributes of an object to be returned. The requester indicates which attributes to return by specifying an "information class". This is an enumeration that the responder understands and translates to a specific structure to return (the structure contains the attributes indicated by the information class).

For example, to retrieve the name of a trusted domain, a requester would specify the information level "TrustedDomainNameInformation" to the LsarQueryTrustedDomainInfo method.

The following are the methods that follow the query pattern:

* + LsarQueryDomainInformationPolicy
  + LsarQueryForestTrustInformation
  + LsarQueryInformationPolicy2
  + LsarQuerySecret
  + LsarQueryTrustedDomainInfo
  + LsarQueryTrustedDomainInfoByName
  + LsarQueryInfoTrustedDomain
* **Set pattern**: This pattern enables specified object attributes to be set. The requester makes a request for which attributes to update by specifying an "information class". Similar to the Query pattern, this information level allows the caller to specify to the responder which attributes are being sent in the request.

The following are the methods that follow the set pattern:

* + LsarSetDomainInformationPolicy
  + LsarSetForestTrustInformation
  + LsarSetInformationPolicy2
  + LsarSetSecret
  + LsarAddPrivilegesToAccount
  + LsarRemovePrivilegesFromAccount
  + LsarAddAccountRights
  + LsarRemoveAccountRights
* **Delete pattern**: This pattern enables a requester to delete a specified object.

The following are the methods that follow the delete pattern:

* + LsarDeleteObject
  + LsarDeleteTrustedDomain
* **Lookup pattern**: This pattern enables a caller to translate between different representations of an entity (in the case of this protocol, names and identifiers of privileges).

The following are the methods that follow the lookup pattern:

* + LsarLookupPrivilegeName
  + LsarLookupPrivilegeValue
  + LsarLookupPrivilegeDisplayName
* **Security pattern**: This pattern enables a caller to specify or query the access control at the level of individual objects.

The following are the methods that follow the security pattern:

* + LsarSetSecurityObject
  + LsarQuerySecurityObject
* **Miscellaneous**: The following method does not fall into a general pattern. A brief description is given here. See the message processing section for details.

LsarClose: This method releases responder resources associated with the RPC context handle that is passed as a parameter.

## Relationship to Other Protocols

The Local Security Authority (Domain Policy) Remote Protocol is composed of a subset of [**opnums**](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7) in an interface that also includes the Local Security Authority (Translation Methods) Remote Protocol [[MS-LSAT]](%5bMS-LSAT%5d.pdf#Section_1ba21e6fd8a9462c91534375f2020894).

The Local Security Authority (Domain Policy) Remote Protocol is dependent on [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331), which is used for communication between [**domain members**](#gt_6234a38c-ed1b-4c69-969f-6e6479566f65) and [**domain controllers**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd).

This protocol shares the **Domain Name** field of the abstract data Account Domain Information, as specified in section [3.1.1.1](#Section_c939c70e8cf04d90a9cc1c5002951723) of this specification, with the Workstation Service Remote Protocol [[MS-WKST]](%5bMS-WKST%5d.pdf#Section_5bb08058bc364d3cabebb132228281b7).

This protocol depends on [**Server Message Block (SMB)**](#gt_09dbec39-5e75-4d9a-babf-1c9f1d499625) protocols for sending messages on the wire.

Authentication protocols like the Kerberos Protocol Extensions [[MS-KILE]](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9) and translation protocols like the Directory Replication Service (DRS) Remote Protocol [[MS-DRSR]](%5bMS-DRSR%5d.pdf#Section_f977faaa673e4f66b9bf48c640241d47) and Local Security Authority (Translation Methods) Remote Protocol [MS-LSAT] depend on the abstract data model introduced by this protocol in section [3.1.1](#Section_0877fdc4184f40b0b378f50d6647d23e). These protocols use the information in the Local Security Authority (Domain Policy) Remote Protocol to locate a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) that can process further requirements on that protocol.

The Active Directory Technical Specification [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) discusses [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90), which is used by this protocol when running on a domain controller.

The server-side protocol relationships for non-domain controller and domain controller configurations are illustrated in the following diagrams.

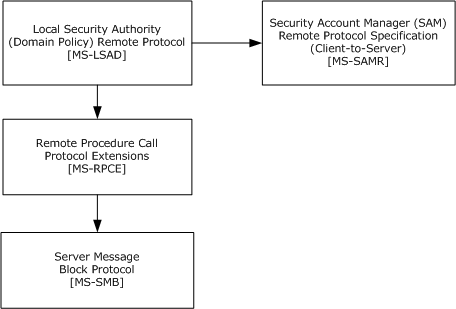


Figure 1: Server-side protocol relationships for a non-domain controller configuration

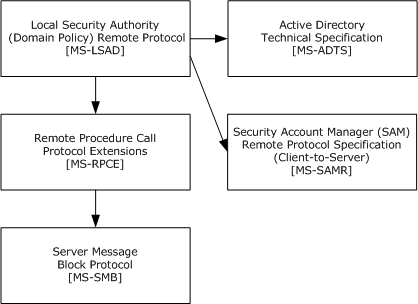


Figure 2: Server-side protocol relationships for a domain controller configuration

## Prerequisites/Preconditions

This protocol has the prerequisites specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) as being common to protocols that depend on [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331).

## Applicability Statement

This protocol is applicable to the following two high-level scenarios:

1. Remote management of [**trusted domains**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4), [**account objects**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b) or [**secret objects**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d), or other miscellaneous machine and [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) policy settings controlled by the protocol.
2. Retrieval of policy settings by clients.

To achieve the first scenario, this entire specification has to be implemented.

To achieve the second scenario, only [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) methods [LsarOpenPolicy2 (section 3.1.4.4.1)](#Section_9456a9637c214710af77d0a2f5a72d6b), [LsarOpenPolicy (section 3.1.4.4.2)](#Section_2a482ccf1f8946938594855ff738ae8a), [LsarQueryInformationPolicy2 (section 3.1.4.4.3)](#Section_516f503c0230489db012e650b46b66a2), [LsarQueryInformationPolicy (section 3.1.4.4.4)](#Section_3564ba7084ea4f04a9dcdede9f96a8bf), and [LsarClose (section 3.1.4.9.4)](#Section_99dd2d7ab0fc4c6d837a2b4d342383ae) (and associated data structures specified in these method definitions) have to be implemented by a listener of this protocol.

Although significant protocol functionality is not dependent on server configuration, some functionality might depend on server configuration. Certain aspects of this protocol might depend on the server being a [**DC**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), including [**PDCs**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d), [**BDCs**](#gt_ce1138c6-7ab4-4c37-98b4-95599071c3c3), [**GC servers**](#gt_a5a99ce4-e206-42dc-8874-e103934c5b0d), and [**RODCs**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870), or on being a non–DC, and also on the server reaching a certain [**forest functional level**](#gt_b3240417-ca43-4901-90ec-fde55b32b3b8). These requirements are explained in their respective message processing sections.

## Versioning and Capability Negotiation

* **Supported transports**: The protocol runs over [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331)-named pipes and TCP/IP, as specified in section [2.1](#Section_64ea7ac432ef44f6ab51ea2b5a1c2390).
* **Protocol version**: This protocol's RPC interface has a single version number, but the interface has been extended by placing additional methods at the end. The use of these methods is specified in section [3.1](#Section_63acb76d473a407dbc0c548118e53055).
* **Structure version**: [LSAPR\_ACL (section 2.2.3.2)](#Section_a9a03a855b084bb581c92c68751693ac) structures are versioned using the first field in the structure. Only one version of those structures is used in this protocol.
* **Localization**: This protocol uses text strings in various functions. Localization considerations for such strings are specified in section [3.1.1.2.1](#Section_1a92af76d45f42c3b67cf1dc61bd6ee1).

## Vendor-Extensible Fields

This protocol uses NTSTATUS values as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.3. Vendors are free to choose their own values for this field, provided that the C bit (0x20000000) is set, which indicates that it is a customer code.

## Standards Assignments

This protocol has no standards assignments. It uses private allocations for the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface [**universally unique identifier (UUID)**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) and the RPC [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee).

| Parameter | Value | Reference |
| --- | --- | --- |
| lsarpc Interface UUID | {12345778-1234-ABCD-EF00-0123456789AB} | [C706] section A.2.5. |
| RPC endpoint | \PIPE\lsarpc | section [2.1](#Section_64ea7ac432ef44f6ab51ea2b5a1c2390) |

# Messages

This section describes the supported transports and details of the messages defined for this protocol.

## Transport

This protocol MUST use [**Server Message Block (SMB)**](#gt_09dbec39-5e75-4d9a-babf-1c9f1d499625) [**RPC protocol sequences**](#gt_0c171cc7-e9c4-41b6-95a9-536db0042c7a).

This protocol MUST use "\PIPE\lsarpc" as the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) when using RPC over SMB.[<1>](#Appendix_A_1" \o "Product behavior note 1)

For authentication and authorization [**services**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a), both the requester and responder of this protocol MUST use the SMB transport to communicate the identity of the requester, as specified in [[MS-SMB]](%5bMS-SMB%5d.pdf#Section_f210069c70864dc2885e861d837df688) section 3.2.4.2.4 and [[MS-SMB2]](%5bMS-SMB2%5d.pdf#Section_5606ad475ee0437a817e70c366052962) section 3.2.4.2.3.

For confidentiality and tamper resistance services, the requester and responder MAY use the functionality provided by the SMB transport, as specified in [MS-SMB] sections 2.2.3.1 and 2.2.4.5.2.1 and [MS-SMB2] sections 2.2.3 and 2.2.4.[<2>](#Appendix_A_2" \o "Product behavior note 2)

The requester MUST NOT use the RPC-provided security-support-provider mechanisms (for authentication, authorization, confidentiality, or tamper-resistance services).[<3>](#Appendix_A_3" \o "Product behavior note 3)

The responder MAY use the RPC-provided security-support-provider mechanisms as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.2.1.4.1.1.[<4>](#Appendix_A_4" \o "Product behavior note 4)

The server SHOULD[<5>](#Appendix_A_5" \o "Product behavior note 5) reject calls that do not use an authentication level of RPC\_C\_AUTHN\_LEVEL\_NONE, RPC\_C\_AUTHN\_LEVEL\_PKT\_INTEGRITY, or RPC\_C\_AUTHN\_LEVEL\_PKT\_PRIVACY ([MS-RPCE] section 2.2.1.1.8).

Cryptographic operations (as specified in section [5.1](#Section_ed292c30f14a4568b5992e69358144d8)) MUST utilize a session key obtained from the SMB session on the client or server.

This protocol MUST use the [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) and version number as follows:

* UUID: See Standards Assignments in section [1.9](#Section_302bc4e169fb48cbabb20d817c8473e5).
* Version number: 0.0.

The security settings used in this protocol vary depending on the role of the [**RPC client**](#gt_e5a7be6b-98db-4e8d-8116-5893f43ab48b) and [**RPC server**](#gt_ae65dac0-cd24-4e83-a946-6d1097b71553), the function being used, and the specific parameters being used. Security settings are therefore specified in message processing sections for each message.

This protocol SHOULD[<6>](#Appendix_A_6" \o "Product behavior note 6) configure RPC to enforce Maximum Server Input Data Size of 1 MB. Additional details are available in [MS-RPCE] section 3.3.3.5.4. This configuration introduces additional restrictions on the upper limits for the sizes of data types defined under section [2.2](#Section_4d4678cf32154ecc8dc95a2aaa0e1eb0) when those data types are used in RPC messages.

## Common Data Types

This protocol MUST indicate to the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) runtime that it is to support both the [**NDR**](#gt_9ebf9540-2c31-43bc-bc56-4a932faabf2d) and [**NDR64**](#gt_29385c51-3799-4da6-8291-32fc46a81970) transfer syntaxes and provide a negotiation mechanism for determining which transfer syntax will be used, as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 12 and in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.1.5.6.

This protocol contains messages with parameters that do not have any effect on message processing in any environment; however, the parameters remain for backward compatibility of the interfaces. These will be called out as ignored in sections on data type definition, message definition, and message processing. These values MUST be ignored on receipt and SHOULD be set to zero when sent, unless specified otherwise.

In addition to RPC base types and definitions specified in [C706] and [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2), other data types are defined in this specification.[<7>](#Appendix_A_7" \o "Product behavior note 7)

The following lists summarize the types defined in this specification.

**Note**  [**LUID**](#gt_96b64af9-1896-4bde-b988-54d469c5affd), NTSTATUS, RPC\_SID, and RPC\_UNICODE\_STRING are specified in [MS-DTYP] sections 2.3.7, 2.2.38, 2.4. 2.4.2.3, and 2.3.10, respectively.

**Note**  The LARGE INTEGER structure, when it represents time in this protocol, is used as a 64-bit value that represents the number of 100-nanosecond intervals since January 1, 1601, [**Coordinated Universal Time (UTC)**](#gt_f2369991-a884-4843-a8fa-1505b6d5ece7).

Constant value definitions:

* [ACCESS\_MASK (section 2.2.1.1)](#Section_7aeb7f170a6e4f04ac7e7b1363cf9ecf)
* [POLICY\_SYSTEM\_ACCESS\_MODE (section 2.2.1.2)](#Section_ba5a83c10ffe4a819e915739274c03db)
* [SECURITY\_INFORMATION (section 2.2.1.3)](#Section_62175da4e30f4c12b1c4dae0434e38af)

Basic data types:

* [LSAPR\_HANDLE (section 2.2.2.1)](#Section_0d093105e8c845f7a79d182aafd60c6e)
* [PLSAPR\_HANDLE (section 2.2.2.2)](#Section_8a7f72c51f5a4b7ea9c6e78b4f430fc7)
* [LSA\_UNICODE\_STRING (section 2.2.2.3)](#Section_4b35e17e405c4e998ebe8b28f047156f)
* [LSAPR\_OBJECT\_ATTRIBUTES (section 2.2.2.4)](#Section_ad9e183d64744641a6d9d3796d2d604b)
* [LSAPR\_SR\_SECURITY\_DESCRIPTOR (section 2.2.2.5)](#Section_5564065e3f3d4481a385367cc9b042c4)

Data types referenced by basic data types:

* [STRING (section 2.2.3.1)](#Section_94a41a4fbd5d4c3eafd4cc17e83a6e01)
* [LSAPR\_ACL (section 2.2.3.2)](#Section_a9a03a855b084bb581c92c68751693ac)
* [SECURITY\_DESCRIPTOR\_CONTROL (section 2.2.3.3)](#Section_c704a67c983641d99b18acd596cc884e)
* [LSAPR\_SECURITY\_DESCRIPTOR (section 2.2.3.4)](#Section_d5cf869d674449cca67730ccb9217def)
* [SECURITY\_IMPERSONATION\_LEVEL (section 2.2.3.5)](#Section_720cea10cee24c459084c6fa7d67d18d)
* [SECURITY\_CONTEXT\_TRACKING\_MODE (section 2.2.3.6)](#Section_6bb42770b92441ff8a5783e37b8b7797)
* [SECURITY\_QUALITY\_OF\_SERVICE (section 2.2.3.7)](#Section_0ddf315053b542a5b0ec518bce67738c)

Policy query/set data types:

* [POLICY\_INFORMATION\_CLASS (section 2.2.4.1)](#Section_9ce0bb37fc6c4230b1097e1881660b83)
* [LSAPR\_POLICY\_INFORMATION (section 2.2.4.2)](#Section_6e63a2c85ddb411aa2539c55afc49834)
* [POLICY\_AUDIT\_LOG\_INFO (section 2.2.4.3)](#Section_3fff1c62e8b14bc8b18a3ba6458ec622)
* [LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO (section 2.2.4.4)](#Section_d00fc364577d4ed0b3a5952d78b67695)
* [LSAPR\_POLICY\_PRIMARY\_DOM\_INFO (section 2.2.4.5)](#Section_0f3f5d3f66d245a08c28ede86f4cd4a8)
* [LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO (section 2.2.4.6)](#Section_bfad54243e2043bd87f6d35b4253792e)
* [LSAPR\_POLICY\_PD\_ACCOUNT\_INFO (section 2.2.4.7)](#Section_b04175b3fedf4dda9034f754a10fe64e)
* [POLICY\_LSA\_SERVER\_ROLE (section 2.2.4.8)](#Section_620010b4b4394d46893acb67246de5fc)
* [POLICY\_LSA\_SERVER\_ROLE\_INFO (section 2.2.4.9)](#Section_d37dbc6504f34db8b40a4e9dd6c12520)
* [LSAPR\_POLICY\_REPLICA\_SRCE\_INFO (section 2.2.4.10)](#Section_fb7df2bb99e7402f833424d47e23ec00)
* [POLICY\_MODIFICATION\_INFO (section 2.2.4.11)](#Section_c80ae9d5d0c14d5ca0ae77eae7bfac25)
* [POLICY\_AUDIT\_FULL\_SET\_INFO (section 2.2.4.12)](#Section_3224400e3c404e64810a8b11341ba4c6)
* [POLICY\_AUDIT\_FULL\_QUERY\_INFO (section 2.2.4.13)](#Section_0ef0845ff20e4897ad2988c0c07be0f4)
* [LSAPR\_POLICY\_DNS\_DOMAIN\_INFO (section 2.2.4.14)](#Section_3e15a02e25d346aa9c608def03c824d2)
* [POLICY\_DOMAIN\_INFORMATION\_CLASS (section 2.2.4.15)](#Section_566a61fc2e9947c899ca62f7e22cb15d)
* [LSAPR\_POLICY\_DOMAIN\_INFORMATION (section 2.2.4.16)](#Section_1a9c523ba67a485f8f8b8fca05ca9334)
* [POLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO (section 2.2.4.17)](#Section_38bd52a04514468fb342d7421a51a316)
* [LSAPR\_POLICY\_DOMAIN\_EFS\_INFO (section 2.2.4.18)](#Section_3ba6e751cf914d87a74c488bb927a54c)
* [LSAPR\_DOMAIN\_KERBEROS\_TICKET\_INFO (section 2.2.4.19)](#Section_afcc492012d348e0ab95a8989ebbd41d)

Account query/set data types:

* [LSAPR\_ACCOUNT\_INFORMATION (section 2.2.5.1)](#Section_98540c1c09cc4ee2934acdde3de0c77f)
* [LSAPR\_ACCOUNT\_ENUM\_BUFFER (section 2.2.5.2)](#Section_727c2d44879448969fba5e1725bc288e)
* [LSAPR\_USER\_RIGHT\_SET (section 2.2.5.3)](#Section_dcaca8ef34a342dd85b698363eb108ff)
* [LSAPR\_LUID\_AND\_ATTRIBUTES (section 2.2.5.4)](#Section_03c834c0f3104e0c832eb6e7688364d1)
* [LSAPR\_PRIVILEGE\_SET (section 2.2.5.5)](#Section_a30a5720778442f4b03ab14f4e486bae)

Secret query/set data types:

* [LSAPR\_CR\_CIPHER\_VALUE (section 2.2.6.1)](#Section_782eda77b82e413487c9eb5e67f18f06)

Trusted domain query/set data types:

* [LSAPR\_TRUST\_INFORMATION (section 2.2.7.1)](#Section_71e86cddae194a029179a2a103b383a0)
* [TRUSTED\_INFORMATION\_CLASS (section 2.2.7.2)](#Section_360691136c3845e8920e17f8ef36f578)
* [LSAPR\_TRUSTED\_DOMAIN\_INFO (section 2.2.7.3)](#Section_65564571dd0d49a98a2a6dba8ab57091)
* [LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO (section 2.2.7.4)](#Section_71c5724f447e452c9cb9a0fd90d88594)
* [LSAPR\_TRUSTED\_CONTROLLERS\_INFO (section 2.2.7.5)](#Section_5382bd8969c646f2beb17b70e5befbc5)
* [TRUSTED\_POSIX\_OFFSET\_INFO (section 2.2.7.6)](#Section_b091ee7ef5c34b4885671b08ea002221)
* [LSAPR\_TRUSTED\_PASSWORD\_INFO (section 2.2.7.7)](#Section_33d7a9e4c9ca40219627337d89e656a3)
* [LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_BASIC (section 2.2.7.8)](#Section_c101591de1b042bd8cc5f8866c3b5757)
* [LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX (section 2.2.7.9)](#Section_f28f42b7173c4cda98093fe4a5213ab3)
* [LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 (section 2.2.7.10)](#Section_dd92d4d9227f4ef1b42bef3f056f8aaa)
* [LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION (section 2.2.7.11)](#Section_084fdb6b5bc349129aed0257159996dd)
* [LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL (section 2.2.7.12)](#Section_3b1c61fe6f074d83af543a381de5c5d1)
* [LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION (section 2.2.7.13)](#Section_9f9feebce9e141c18c4802f83a227a14)
* [LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL (section 2.2.7.14)](#Section_2e9e2c847b004fb18de588d4cfedd2b3)
* [LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2 (section 2.2.7.15)](#Section_e529d0945de44738adc4efa1a7d1106f)
* [LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB (section 2.2.7.16)](#Section_da8f32a10a164194810d06cc0698595a)
* [LSAPR\_AUTH\_INFORMATION (section 2.2.7.17)](#Section_cedb0d1bc7c0448099fc279b06f22a0c)
* [TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES (section 2.2.7.18)](#Section_7c519a643dc14be6a17d76817cff6e39)
* [LSAPR\_TRUSTED\_ENUM\_BUFFER (section 2.2.7.19)](#Section_78f8a2e44f3d40f5bdd19dacdf1c832c)
* [LSAPR\_TRUSTED\_ENUM\_BUFFER\_EX (section 2.2.7.20)](#Section_b9b29ed6786e483f9e3b776eb014086b)
* [LSA\_FOREST\_TRUST\_RECORD (section 2.2.7.21)](#Section_08cf1a65ef7c46d3aa4d558f5135df3d)
* [LSA\_FOREST\_TRUST\_RECORD\_TYPE (section 2.2.7.22)](#Section_700a91e8a1a44e1b9ad6096b3cf0bef0)
* [LSA\_FOREST\_TRUST\_BINARY\_DATA (section 2.2.7.23)](#Section_d4859b44b7764237baa112dc28c19634)
* [LSA\_FOREST\_TRUST\_DOMAIN\_INFO (section 2.2.7.24)](#Section_451ac72fe9ad4a4f961fd04a2a5b1515)
* [LSA\_FOREST\_TRUST\_INFORMATION (section 2.2.7.25)](#Section_2993ffabc0c846439a794ff7d31922dc)
* [LSA\_FOREST\_TRUST\_COLLISION\_RECORD\_TYPE (section 2.2.7.26)](#Section_afc7d769a31748059f4585d5393b57af)
* [LSA\_FOREST\_TRUST\_COLLISION\_RECORD (section 2.2.7.27)](#Section_32178d2cca744f538264af1906f95011)
* [LSA\_FOREST\_TRUST\_COLLISION\_INFORMATION (section 2.2.7.28)](#Section_db0e91319a194cb5937a0e3a5767a0b2)

Privilege data types:

* [LSAPR\_POLICY\_PRIVILEGE\_DEF (section 2.2.8.1)](#Section_f36d47375b2f4bc08f29e7b4c71b7401)
* [LSAPR\_PRIVILEGE\_ENUM\_BUFFER (section 2.2.8.2)](#Section_c0278280b4b64538b3aaeb40f64f42fb)

The following citation contains a timeline of when each structure, data type, or enumeration was introduced.[<8>](#Appendix_A_8" \o "Product behavior note 8)

### Constant Value Definitions

#### ACCESS\_MASK

The ACCESS\_MASK data type is a bitmask that defines the user rights that an object is to be granted. Access types are reconciled with the [**discretionary access control list (DACL)**](#gt_d727f612-7a45-48e4-9d87-71735d62b321) of the object to determine whether the access requested is assigned or denied.

The ACCESS\_MASK data type is defined in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.3. The following declaration is an alternative definition.

This type is declared as follows:

1. typedef unsigned long ACCESS\_MASK;

##### ACCESS\_MASK for All Objects

Certain [ACCESS\_MASK](#Section_7aeb7f170a6e4f04ac7e7b1363cf9ecf) flags apply equally to all types of objects. These flags are described in the following table.

| Value | Meaning |
| --- | --- |
| DELETE  0x00010000 | Delete object. |
| READ\_CONTROL  0x00020000 | The read value of a [**DACL**](#gt_d727f612-7a45-48e4-9d87-71735d62b321) and owner in a [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350). |
| WRITE\_DAC  0x00040000 | The write value of a DACL in a security descriptor. |
| WRITE\_OWNER  0x00080000 | The write value of the owner in a security descriptor. |
| MAXIMUM\_ALLOWED  0x02000000 | Used in requesting access; get as much access as the server will allow. |

The four high-order bits in ACCESS\_MASK values are translated by the responder into specific ACCESS\_MASK values using the following tables, depending on the type of the object that the operation is performed on. For numeric values of the symbolic names used in these tables, refer to section [2.2.1.1.2](#Section_b61b7268987a420b84f96c75f8dc8558) for policy objects, section [2.2.1.1.3](#Section_fc3b5e24b1a24c7983d7256ceaef8ff4) for [**account objects**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b), section [2.2.1.1.4](#Section_88c6bd186c404a82ae19fe7bfec5108b) for [**secret objects**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d), and section [2.2.1.1.5](#Section_e035f552031348b79bcafdd9fd4e948e) for [**trusted domain objects**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4). In the following tables, the symbol '|' is used to indicate that the value represented by the symbol is to be logically combined by using the bitwise OR operation with the other operant.

| ACCESS\_MASK value to be translated | Translated to when used with policy object |
| --- | --- |
| 0x80000000 | POLICY\_VIEW\_AUDIT\_INFORMATION | POLICY\_GET\_PRIVATE\_INFORMATION | READ\_CONTROL  0x00020006 |
| 0x40000000 | POLICY\_TRUST\_ADMIN | POLICY\_CREATE\_ACCOUNT | POLICY\_CREATE\_SECRET | POLICY\_CREATE\_PRIVILEGE | POLICY\_SET\_DEFAULT\_QUOTA\_LIMITS | POLICY\_SET\_AUDIT\_REQUIREMENTS | POLICY\_AUDIT\_LOG\_ADMIN | POLICY\_SERVER\_ADMIN | READ\_CONTROL  0x000207F8 |
| 0x20000000 | POLICY\_VIEW\_LOCAL\_INFORMATION | POLICY\_LOOKUP\_NAMES | READ\_CONTROL  0x00020801 |
| 0x10000000 | POLICY\_VIEW\_LOCAL\_INFORMATION | POLICY\_VIEW\_AUDIT\_INFORMATION | POLICY\_GET\_PRIVATE\_INFORMATION | POLICY\_TRUST\_ADMIN | POLICY\_CREATE\_ACCOUNT | POLICY\_CREATE\_SECRET | POLICY\_CREATE\_PRIVILEGE | POLICY\_SET\_DEFAULT\_QUOTA\_LIMITS | POLICY\_SET\_AUDIT\_REQUIREMENTS | POLICY\_AUDIT\_LOG\_ADMIN | POLICY\_SERVER\_ADMIN | POLICY\_LOOKUP\_NAMES | DELETE | READ\_CONTROL | WRITE\_DAC | WRITE\_OWNER  0x000F0FFF |

| ACCESS\_MASK value to be translated | Translated to when used with account object |
| --- | --- |
| 0x80000000 | ACCOUNT\_VIEW | READ\_CONTROL  0x00020001 |
| 0x40000000 | ACCOUNT\_ADJUST\_PRIVILEGES | ACCOUNT\_ADJUST\_QUOTAS | ACCOUNT\_ADJUST\_SYSTEM\_ACCESS | READ\_CONTROL  0x0002000E |
| 0x20000000 | READ\_CONTROL  0x00020000 |
| 0x10000000 | ACCOUNT\_VIEW | ACCOUNT\_ADJUST\_PRIVILEGES | ACCOUNT\_ADJUST\_QUOTAS | ACCOUNT\_ADJUST\_SYSTEM\_ACCESS | DELETE | READ\_CONTROL | WRITE\_DAC | WRITE\_OWNER  0x000F000F |

| ACCESS\_MASK value to be translated | Translated to when used with secret object |
| --- | --- |
| 0x80000000 | SECRET\_QUERY\_VALUE | READ\_CONTROL  0x00020002 |
| 0x40000000 | SECRET\_SET\_VALUE | READ\_CONTROL  0x00020001 |
| 0x20000000 | READ\_CONTROL  0x00020000 |
| 0x10000000 | SECRET\_QUERY\_VALUE | SECRET\_SET\_VALUE | DELETE | READ\_CONTROL | WRITE\_DAC | WRITE\_OWNER  0x000F0003 |

| ACCESS\_MASK value to be translated | Translated to when used with trusted domain object |
| --- | --- |
| 0x80000000 | TRUSTED\_QUERY\_DOMAIN\_NAME | READ\_CONTROL  0x00020001 |
| 0x40000000 | TRUSTED\_SET\_CONTROLLERS | TRUSTED\_SET\_POSIX | READ\_CONTROL  0x00020014 |
| 0x20000000 | TRUSTED\_QUERY\_CONTROLLERS | TRUSTED\_QUERY\_POSIX | READ\_CONTROL  0x0002000A |
| 0x10000000 | TRUSTED\_QUERY\_DOMAIN\_NAME | TRUSTED\_QUERY\_CONTROLLERS | TRUSTED\_SET\_CONTROLLERS | TRUSTED\_QUERY\_POSIX | TRUSTED\_SET\_POSIX | TRUSTED\_SET\_AUTH | TRUSTED\_QUERY\_AUTH | DELETE | READ\_CONTROL | WRITE\_DAC | WRITE\_OWNER  0x000F007F |

##### ACCESS\_MASK for Policy Objects

The following [ACCESS\_MASK](#Section_7aeb7f170a6e4f04ac7e7b1363cf9ecf) flags apply to policy objects.

| Value | Meaning |
| --- | --- |
| 0x00000000 | No access. |
| POLICY\_VIEW\_LOCAL\_INFORMATION  0x00000001 | Access to view local information. |
| POLICY\_VIEW\_AUDIT\_INFORMATION  0x00000002 | Access to view audit information. |
| POLICY\_GET\_PRIVATE\_INFORMATION  0x00000004 | Access to view private information. |
| POLICY\_TRUST\_ADMIN  0x00000008 | Access to administer [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) relationships. |
| POLICY\_CREATE\_ACCOUNT  0x00000010 | Access to create [**account objects**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b). |
| POLICY\_CREATE\_SECRET  0x00000020 | Access to create [**secret objects**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d). |
| POLICY\_CREATE\_PRIVILEGE  0x00000040 | Access to create [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940).  **Note**  New privilege creation is not currently a part of the protocol, so this flag is not actively used. |
| POLICY\_SET\_DEFAULT\_QUOTA\_LIMITS  0x00000080 | Access to set default quota limits.  **Note**  Quota limits are not currently a part of the protocol, so this flag is not actively used. |
| POLICY\_SET\_AUDIT\_REQUIREMENTS  0x00000100 | Access to set audit requirements. |
| POLICY\_AUDIT\_LOG\_ADMIN  0x00000200 | Access to administer the audit log. |
| POLICY\_SERVER\_ADMIN  0x00000400 | Access to administer policy on the server. |
| POLICY\_LOOKUP\_NAMES  0x00000800 | Access to translate names and [**security identifiers (SIDs)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d). |
| POLICY\_NOTIFICATION  0x00001000 | Access to be notified of policy changes.[<9>](#Appendix_A_9" \o "Product behavior note 9) |

##### ACCESS\_MASK for Account Objects

The following [ACCESS\_MASK](#Section_7aeb7f170a6e4f04ac7e7b1363cf9ecf) flags apply to [**account objects**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b).

| Value | Meaning |
| --- | --- |
| ACCOUNT\_VIEW  0x00000001 | View account information. |
| ACCOUNT\_ADJUST\_PRIVILEGES  0x00000002 | Change [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) on an account. |
| ACCOUNT\_ADJUST\_QUOTAS  0x00000004 | Change quotas on an account. |
| ACCOUNT\_ADJUST\_SYSTEM\_ACCESS  0x00000008 | Change system access. |

##### ACCESS\_MASK for Secret Objects

The following [ACCESS\_MASK](#Section_7aeb7f170a6e4f04ac7e7b1363cf9ecf) flags apply to [**secret objects**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d).

| Value | Meaning |
| --- | --- |
| SECRET\_SET\_VALUE  0x00000001 | Set secret value. |
| SECRET\_QUERY\_VALUE  0x00000002 | Query secret value. |

##### ACCESS\_MASK for Trusted Domain Objects

The following [ACCESS\_MASK](#Section_7aeb7f170a6e4f04ac7e7b1363cf9ecf) flags apply to [**trusted domain objects**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4).[<10>](#Appendix_A_10" \o "Product behavior note 10)

| Value | Meaning |
| --- | --- |
| TRUSTED\_QUERY\_DOMAIN\_NAME  0x00000001 | View [**domain name**](#gt_45a1c9f1-0263-49a8-97c7-7aca1a99308c) information. |
| TRUSTED\_QUERY\_CONTROLLERS  0x00000002 | View "Domain Controllers" information. |
| TRUSTED\_SET\_CONTROLLERS  0x00000004 | Change "Domain Controllers" information. |
| TRUSTED\_QUERY\_POSIX  0x00000008 | View POSIX information. |
| TRUSTED\_SET\_POSIX  0x00000010 | Change POSIX information. |
| TRUSTED\_SET\_AUTH  0x00000020 | Change authentication information. |
| TRUSTED\_QUERY\_AUTH  0x00000040 | View authentication information. |

#### POLICY\_SYSTEM\_ACCESS\_MODE

The POLICY\_SYSTEM\_ACCESS\_MODE data type determines the way in which a user (member of a group or alias) is allowed to access the system. All values can be combined in any way by using bitwise OR operations.

| Value | Meaning |
| --- | --- |
| 0x00000000 | No access  The user is not granted any access to the system. |
| POLICY\_MODE\_INTERACTIVE  0x00000001 | The user can logon locally to the system. |
| POLICY\_MODE\_NETWORK  0x00000002 | The user can logon to the system over the network. |
| POLICY\_MODE\_BATCH  0x00000004 | The user can logon to the system as a batch job. |
| 0x00000008 | Reserved |
| POLICY\_MODE\_SERVICE  0x00000010 | The user can logon to the system as a [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a). |
| 0x00000020 | Reserved |
| POLICY\_MODE\_DENY\_INTERACTIVE  0x00000040 | The user is denied the right to interactively logon to the system. This setting supersedes POLICY\_MODE\_INTERACTIVE. |
| POLICY\_MODE\_DENY\_NETWORK  0x00000080 | The user is denied the right to logon to the system from the network. This setting supersedes POLICY\_MODE\_NETWORK. |
| POLICY\_MODE\_DENY\_BATCH  0x00000100 | The user is denied the right to logon to the system as a batch job. This setting supersedes POLICY\_MODE\_BATCH. |
| POLICY\_MODE\_DENY\_SERVICE  0x00000200 | The user is denied the right to logon to the system as a service. This setting supersedes POLICY\_MODE\_SERVICE. |
| POLICY\_MODE\_REMOTE\_INTERACTIVE  0x00000400 | The user can logon to the system as a Remote Desktop client. |
| POLICY\_MODE\_DENY\_REMOTE\_INTERACTIVE  0x00000800 | The user is denied the right to logon to the system as a Remote Desktop client. |
| POLICY\_MODE\_ALL  0x00000FF7 | This flag indicates all allowed bits.[<11>](#Appendix_A_11" \o "Product behavior note 11) |
| POLICY\_MODE\_ALL\_NT4  0x00000037 | This flag indicates all allowed bits.[<12>](#Appendix_A_12" \o "Product behavior note 12) |

The following citation contains a timeline of when each mode was introduced.[<13>](#Appendix_A_13" \o "Product behavior note 13)

#### SECURITY\_INFORMATION

The SECURITY\_INFORMATION type is used to specify which portions of a [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) the caller would like to retrieve or set on an object.

The SECURITY\_INFORMATION data type is defined in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.7. The following declaration is an alternative definition.

This type is declared as follows:

1. typedef unsigned long SECURITY\_INFORMATION;

The following table defines the bits that are relevant to the Local Security Authority (Domain Policy) Remote Protocol.

| Value | Meaning |
| --- | --- |
| OWNER\_SECURITY\_INFORMATION  0x00000001 | Return the Owner portion of the security descriptor. |
| GROUP\_SECURITY\_INFORMATION  0x00000002 | Return the Group portion of the security descriptor. |
| DACL\_SECURITY\_INFORMATION  0x00000004 | Return the [**DACL**](#gt_d727f612-7a45-48e4-9d87-71735d62b321) portion of the security descriptor. |
| SACL\_SECURITY\_INFORMATION  0x00000008 | Return the [**SACL**](#gt_c189801e-3752-4715-88f4-17804dad5782) portion of the security descriptor. |

Other values SHOULD NOT be set.

The server honors the request to set or retrieve security information only if the caller has the appropriate rights to the object.

The following table lists the SECURITY\_INFORMATION bits and the corresponding user rights required of the caller requesting to query information.

| Security information access requested | Rights required of caller on server | Privileges required of caller on server |
| --- | --- | --- |
| OWNER\_SECURITY\_INFORMATION | READ\_CONTROL | Does not apply. |
| GROUP\_SECURITY\_INFORMATION | READ\_CONTROL | Does not apply. |
| DACL\_SECURITY\_INFORMATION | READ\_CONTROL | Does not apply. |
| SACL\_SECURITY\_INFORMATION | Does not apply. | Security [**privilege**](#gt_d8092e10-b227-4b44-b015-511bb8178940). |

The following table lists the SECURITY\_INFORMATION bits and the corresponding user rights required of the caller requesting to set information.

| Security information access requested | Rights required of caller on server | Privileges required of caller on server |
| --- | --- | --- |
| OWNER\_SECURITY\_INFORMATION | WRITE\_OWNER | Take ownership privilege.  **Note**  Either the access bit or the privilege is sufficient; the caller does not need both. |
| GROUP\_SECURITY\_INFORMATION | WRITE\_OWNER | Take-ownership privilege. |
| DACL\_SECURITY\_INFORMATION | WRITE\_DAC | Does not apply. |
| SACL\_SECURITY\_INFORMATION | Does not apply. | Security privilege. |

### Basic Data Types

#### LSAPR\_HANDLE

The LSAPR\_HANDLE type defines a context handle (as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 6) to the target server.

This type is declared as follows:

1. typedef [context\_handle] void\* LSAPR\_HANDLE;

**Note**  For information about the relevance of the context\_handle attribute in this data type, see section [3.1.1.7](#Section_1011130b0413460d81edd1821d141639).

#### PLSAPR\_HANDLE

The PLSAPR\_HANDLE type defines a pointer to a context handle (as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 6).

This type is declared as follows:

1. typedef LSAPR\_HANDLE\* PLSAPR\_HANDLE;

#### LSA\_UNICODE\_STRING

The LSA\_UNICODE\_STRING type is identical to RPC\_UNICODE\_STRING, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.3.10.

This type is declared as follows:

1. typedef RPC\_UNICODE\_STRING LSA\_UNICODE\_STRING, \*PLSA\_UNICODE\_STRING;

#### LSAPR\_OBJECT\_ATTRIBUTES

The LSAPR\_OBJECT\_ATTRIBUTES structure specifies an object and its properties. This structure MUST be ignored except for the **RootDirectory** field, which MUST be NULL.[<14>](#Appendix_A_14" \o "Product behavior note 14)

1. typedef struct \_LSAPR\_OBJECT\_ATTRIBUTES {
2. unsigned long Length;
3. unsigned char\* RootDirectory;
4. PSTRING ObjectName;
5. unsigned long Attributes;
6. PLSAPR\_SECURITY\_DESCRIPTOR SecurityDescriptor;
7. PSECURITY\_QUALITY\_OF\_SERVICE SecurityQualityOfService;
8. } LSAPR\_OBJECT\_ATTRIBUTES,
9. \*PLSAPR\_OBJECT\_ATTRIBUTES;

**Length:**  The length of the structure, in bytes. This field is not used and MUST be ignored.

**RootDirectory:**  This field is not used and MUST be NULL.

**ObjectName:**  A pointer to a STRING structure that contains the object name. This field MUST be ignored. The content is unspecified and no requirements are placed on its value because it is never used.

**Attributes:**  This field MUST be ignored. The content is unspecified and no requirements are placed on its value because it is never used.

**SecurityDescriptor:**  This field contains the security attributes of the object. This field MUST be ignored. The content is unspecified and no requirements are placed on its value because it is never used.

**SecurityQualityOfService:**  This field MUST be ignored. The content is unspecified and no requirements are placed on its value because it is never used.

#### LSAPR\_SR\_SECURITY\_DESCRIPTOR

The LSAPR\_SR\_SECURITY\_DESCRIPTOR structure is used to communicate a self-relative [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350), as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.6.

1. typedef struct \_LSAPR\_SR\_SECURITY\_DESCRIPTOR {
2. [range(0, 262144)] unsigned long Length;
3. [size\_is(Length)] unsigned char\* SecurityDescriptor;
4. } LSAPR\_SR\_SECURITY\_DESCRIPTOR,
5. \*PLSAPR\_SR\_SECURITY\_DESCRIPTOR;

**Length:**  The count of bytes in SecurityDescriptor.[<15>](#Appendix_A_15" \o "Product behavior note 15)

**SecurityDescriptor:**  The contiguous buffer containing the self-relative security descriptor. This field MUST contain the **Length** number of bytes. If the **Length** field has a value other than 0, this field MUST NOT be NULL.

### Data Types Referenced by Basic Data Types

#### STRING

The STRING structure holds a counted string encoded in the [**OEM code page**](#gt_442ab13f-d2c1-4128-b1db-f3bea4b8224e).

This structure has no effect on message processing in any environment.

1. typedef struct \_STRING {
2. unsigned short Length;
3. unsigned short MaximumLength;
4. [size\_is(MaximumLength), length\_is(Length)]
5. char\* Buffer;
6. } STRING,
7. \*PSTRING;

**Length:**  The length, in bytes, of the string pointed to by the **Buffer** member, not including the terminating null character (if any).

**MaximumLength:**  This field contains the total number of bytes in the **Buffer** field.

**Buffer:**  A pointer to the actual string. If **Length** is greater than 0, this field MUST contain a non-NULL value. If **Length** is 0, this field MUST be ignored.

#### LSAPR\_ACL

The LSAPR\_ACL structure defines the header of an [**access control list (ACL)**](#gt_9f92aa05-dd0a-45f2-88d6-89f1fb654395) that specifies a list of security protections applied to an object.

This structure has no effect on message processing in any environment.

1. typedef struct \_LSAPR\_ACL {
2. unsigned char AclRevision;
3. unsigned char Sbz1;
4. unsigned short AclSize;
5. [size\_is(AclSize - 4)] unsigned char Dummy1[\*];
6. } LSAPR\_ACL,
7. \*PLSAPR\_ACL;

**AclRevision:**  The revision level of the LSAPR\_ACL structure. This field MUST be ignored. The content is unspecified, and no requirements are placed on its value because it is never used.

**Sbz1:**  This field is used for alignment. This field MUST be ignored. The content is unspecified, and no requirements are placed on its value because it is never used.

**AclSize:**  The size of this structure in bytes, including the size of the variable sized **Dummy1** field.

**Dummy1:**  This field MUST be ignored. The content is unspecified, and no requirements are placed on its value because it is never used.

The ACL structure is specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.5.

#### SECURITY\_DESCRIPTOR\_CONTROL

The SECURITY\_DESCRIPTOR\_CONTROL type contains a set of bit flags that qualify the meaning of a [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) or its components.

This type has no effect on message processing in any environment.

This type is declared as follows:

1. typedef unsigned short SECURITY\_DESCRIPTOR\_CONTROL, \*PSECURITY\_DESCRIPTOR\_CONTROL;

The flags that are used with this type are as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.6, under the **Control** member of the SECURITY\_DESCRIPTOR structure.

#### LSAPR\_SECURITY\_DESCRIPTOR

The LSAPR\_SECURITY\_DESCRIPTOR structure defines an object's [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350).

This structure has no effect on message processing in any environment.

1. typedef struct \_LSAPR\_SECURITY\_DESCRIPTOR {
2. unsigned char Revision;
3. unsigned char Sbz1;
4. SECURITY\_DESCRIPTOR\_CONTROL Control;
5. PRPC\_SID Owner;
6. PRPC\_SID Group;
7. PLSAPR\_ACL Sacl;
8. PLSAPR\_ACL Dacl;
9. } LSAPR\_SECURITY\_DESCRIPTOR,
10. \*PLSAPR\_SECURITY\_DESCRIPTOR;

**Revision:**  The security descriptor revision level. This field MUST be ignored. The content is unspecified, and no requirements are placed on its value because it is never used.

**Sbz1:**  This field is used for alignment. This field MUST be ignored. The content is unspecified, and no requirements are placed on its value because it is never used.

**Control:**  A set of flags (as specified in section [2.2.3.3](#Section_c704a67c983641d99b18acd596cc884e)) that qualify the meaning of the security descriptor or its individual fields.

**Owner:**  A pointer to the RPC\_SID structure that represents an object's owner as a [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d).

**Group:**  A pointer to the RPC\_SID structure that represents an object's primary group as a SID.

**Sacl:**  A pointer to an [**ACL**](#gt_9f92aa05-dd0a-45f2-88d6-89f1fb654395) structure (as specified in [2.2.3.2](#Section_a9a03a855b084bb581c92c68751693ac)) that contains a [**system access control list (SACL)**](#gt_c189801e-3752-4715-88f4-17804dad5782).

**Dacl:**  A pointer to an ACL structure that contains a [**discretionary access control list (DACL)**](#gt_d727f612-7a45-48e4-9d87-71735d62b321).

The SECURITY\_DESCRIPTOR structure is specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.6.

#### SECURITY\_IMPERSONATION\_LEVEL

The SECURITY\_IMPERSONATION\_LEVEL enumeration defines a set of values that specifies security impersonation levels. These levels govern the degree to which a server process can act on behalf of a client process.

This enumeration has no effect on message processing in any environment.

1. typedef enum \_SECURITY\_IMPERSONATION\_LEVEL
2. {
3. SecurityAnonymous = 0,
4. SecurityIdentification = 1,
5. SecurityImpersonation = 2,
6. SecurityDelegation = 3
7. } SECURITY\_IMPERSONATION\_LEVEL,
8. \*PSECURITY\_IMPERSONATION\_LEVEL;

**SecurityAnonymous:** The server cannot obtain information about the client and cannot impersonate the client.

**SecurityIdentification:** The server can obtain information such as [**security identifiers**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) and [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940), but the server cannot impersonate the client.

**SecurityImpersonation:** The server can impersonate the client's security context on its local system, but cannot impersonate the client when communicating with [**services**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) on remote systems.

**SecurityDelegation:** The server can impersonate the client's security context when communicating with services on remote systems.

#### SECURITY\_CONTEXT\_TRACKING\_MODE

The SECURITY\_CONTEXT\_TRACKING\_MODE type specifies whether the server is to be given a snapshot of the client's security context (called "static tracking") or is to be continually updated to track changes to the client's security context (called "dynamic tracking").

This structure has no effect on message processing in any environment and SHOULD be ignored.

This type is declared as follows:

1. typedef unsigned char SECURITY\_CONTEXT\_TRACKING\_MODE, \*PSECURITY\_CONTEXT\_TRACKING\_MODE;

The following values are possible.

| Value | Meaning |
| --- | --- |
| 0x00 | The server is given a snapshot of the client's security context. |
| 0x01 | The server is continually updated with changes. |

All other values SHOULD be ignored.

#### SECURITY\_QUALITY\_OF\_SERVICE

The SECURITY\_QUALITY\_OF\_SERVICE structure defines information used to support client impersonation.

This structure has no effect on message processing in any environment.

1. typedef struct \_SECURITY\_QUALITY\_OF\_SERVICE {
2. unsigned long Length;
3. SECURITY\_IMPERSONATION\_LEVEL ImpersonationLevel;
4. SECURITY\_CONTEXT\_TRACKING\_MODE ContextTrackingMode;
5. unsigned char EffectiveOnly;
6. } SECURITY\_QUALITY\_OF\_SERVICE,
7. \*PSECURITY\_QUALITY\_OF\_SERVICE;

**Length:**  This value MUST be ignored. No requirements are placed on its value because it is never used.

**ImpersonationLevel:**  This field contains information (as specified in section [2.2.3.5](#Section_720cea10cee24c459084c6fa7d67d18d)) given to the server about the client that describes how the server can represent, or impersonate, the client.

**ContextTrackingMode:**  This field specifies how the server tracks changes to the client's security context (as specified in section [2.2.3.6](#Section_6bb42770b92441ff8a5783e37b8b7797)).

**EffectiveOnly:**  This field specifies whether the server can enable or disable [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) and groups that the client's security context might include. This value MUST be TRUE (nonzero) if the server has this right; otherwise, it MUST be FALSE (0).

### Policy Query/Set Data Types

#### POLICY\_INFORMATION\_CLASS

The POLICY\_INFORMATION\_CLASS enumeration type contains values that specify the type of policy being queried or set by the client.

1. typedef enum \_POLICY\_INFORMATION\_CLASS
2. {
3. PolicyAuditLogInformation = 1,
4. PolicyAuditEventsInformation,
5. PolicyPrimaryDomainInformation,
6. PolicyPdAccountInformation,
7. PolicyAccountDomainInformation,
8. PolicyLsaServerRoleInformation,
9. PolicyReplicaSourceInformation,
10. PolicyInformationNotUsedOnWire,
11. PolicyModificationInformation,
12. PolicyAuditFullSetInformation,
13. PolicyAuditFullQueryInformation,
14. PolicyDnsDomainInformation,
15. PolicyDnsDomainInformationInt,
16. PolicyLocalAccountDomainInformation,
17. PolicyLastEntry
18. } POLICY\_INFORMATION\_CLASS,
19. \*PPOLICY\_INFORMATION\_CLASS;

**PolicyAuditLogInformation:** Information about audit log.

**PolicyAuditEventsInformation:** Auditing options.

**PolicyPrimaryDomainInformation:** Primary domain information.

**PolicyPdAccountInformation:** Obsolete information class.

**PolicyAccountDomainInformation:** Account domain information.

**PolicyLsaServerRoleInformation:** Server role information.

**PolicyReplicaSourceInformation:** Replica source information.

**PolicyInformationNotUsedOnWire:** This enumeration value does not appear on the wire.

**PolicyModificationInformation:** Obsolete information class.

**PolicyAuditFullSetInformation:** Obsolete information class.

**PolicyAuditFullQueryInformation:** Audit log state.

**PolicyDnsDomainInformation:** DNS domain information.

**PolicyDnsDomainInformationInt:** DNS domain information.

**PolicyLocalAccountDomainInformation:** [**Local account domain**](#gt_5127d055-89b1-49ba-adf0-70470d9b9da0) information.

**PolicyLastEntry:** Not used in this protocol. Present to mark the end of the enumeration.

The following citation contains a timeline of when each enumeration value was introduced.[<16>](#Appendix_A_16" \o "Product behavior note 16)

The values in this enumeration are used to define the contents of the [LSAPR\_POLICY\_INFORMATION (section 2.2.4.2)](#Section_6e63a2c85ddb411aa2539c55afc49834) union, where the structure associated with each enumeration value is specified. The structure associated with each enumeration value defines the meaning of that value to this protocol.

#### LSAPR\_POLICY\_INFORMATION

The LSAPR\_POLICY\_INFORMATION union is defined as follows, where the structure depends on the [POLICY\_INFORMATION\_CLASS](#Section_9ce0bb37fc6c4230b1097e1881660b83) specified in this message.

1. typedef
2. [switch\_type(POLICY\_INFORMATION\_CLASS)]
3. union \_LSAPR\_POLICY\_INFORMATION {
4. [case(PolicyAuditLogInformation)]
5. POLICY\_AUDIT\_LOG\_INFO PolicyAuditLogInfo;
6. [case(PolicyAuditEventsInformation)]
7. LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO PolicyAuditEventsInfo;
8. [case(PolicyPrimaryDomainInformation)]
9. LSAPR\_POLICY\_PRIMARY\_DOM\_INFO PolicyPrimaryDomainInfo;
10. [case(PolicyAccountDomainInformation)]
11. LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO PolicyAccountDomainInfo;
12. [case(PolicyPdAccountInformation)]
13. LSAPR\_POLICY\_PD\_ACCOUNT\_INFO PolicyPdAccountInfo;
14. [case(PolicyLsaServerRoleInformation)]
15. POLICY\_LSA\_SERVER\_ROLE\_INFO PolicyServerRoleInfo;
16. [case(PolicyReplicaSourceInformation)]
17. LSAPR\_POLICY\_REPLICA\_SRCE\_INFO PolicyReplicaSourceInfo;
18. [case(PolicyModificationInformation)]
19. POLICY\_MODIFICATION\_INFO PolicyModificationInfo;
20. [case(PolicyAuditFullSetInformation)]
21. POLICY\_AUDIT\_FULL\_SET\_INFO PolicyAuditFullSetInfo;
22. [case(PolicyAuditFullQueryInformation)]
23. POLICY\_AUDIT\_FULL\_QUERY\_INFO PolicyAuditFullQueryInfo;
24. [case(PolicyDnsDomainInformation)]
25. LSAPR\_POLICY\_DNS\_DOMAIN\_INFO PolicyDnsDomainInfo;
26. [case(PolicyDnsDomainInformationInt)]
27. LSAPR\_POLICY\_DNS\_DOMAIN\_INFO PolicyDnsDomainInfoInt;
28. [case(PolicyLocalAccountDomainInformation)]
29. LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO PolicyLocalAccountDomainInfo;
30. } LSAPR\_POLICY\_INFORMATION,
31. \*PLSAPR\_POLICY\_INFORMATION;

#### POLICY\_AUDIT\_LOG\_INFO

The POLICY\_AUDIT\_LOG\_INFO structure contains information about the state of the audit log. The following structure corresponds to the PolicyAuditLogInformation information class.

1. typedef struct \_POLICY\_AUDIT\_LOG\_INFO {
2. unsigned long AuditLogPercentFull;
3. unsigned long MaximumLogSize;
4. LARGE\_INTEGER AuditRetentionPeriod;
5. unsigned char AuditLogFullShutdownInProgress;
6. LARGE\_INTEGER TimeToShutdown;
7. unsigned long NextAuditRecordId;
8. } POLICY\_AUDIT\_LOG\_INFO,
9. \*PPOLICY\_AUDIT\_LOG\_INFO;

**AuditLogPercentFull:**  A measure of how full the audit log is, as a percentage.

**MaximumLogSize:**  The maximum size of the auditing log, in kilobytes (KB).

**AuditRetentionPeriod:**  The auditing log retention period (64-bit signed integer), a 64-bit value that represents the number of 100-nanosecond intervals since January 1, 1601, [**UTC**](#gt_f2369991-a884-4843-a8fa-1505b6d5ece7). An audit record can be discarded if its time stamp predates the current time minus the retention period.

**AuditLogFullShutdownInProgress:**  A Boolean flag; indicates whether or not a system shutdown is being initiated due to the security audit log becoming full. This condition occurs only if the system is configured to shut down when the log becomes full.

After a shutdown has been initiated, this flag MUST be set to TRUE (nonzero). If an administrator can correct the situation before the shutdown becomes irreversible, this flag MUST be reset to FALSE (0).

This field MUST be ignored for set operations.

**TimeToShutdown:**  A 64-bit value that represents the number of 100-nanosecond intervals since January 1, 1601, UTC. If the AuditLogFullShutdownInProgress flag is set, this field MUST contain the time left before the shutdown becomes irreversible.

**NextAuditRecordId:**  Not in use. This field SHOULD be set to zero when sent, and MUST be ignored on receipt.

#### LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO

The LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO structure contains auditing options on the server.

1. typedef struct \_LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO {
2. unsigned char AuditingMode;
3. [size\_is(MaximumAuditEventCount)]
4. unsigned long\* EventAuditingOptions;
5. [range(0,1000)] unsigned long MaximumAuditEventCount;
6. } LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO,
7. \*PLSAPR\_POLICY\_AUDIT\_EVENTS\_INFO;

**AuditingMode:**  0 indicates that auditing is disabled. All other values indicate that auditing is enabled.

**EventAuditingOptions:**  An array of values specifying the auditing options for a particular audit type. The auditing type of an element is represented by its index in the array, which is identified by the [POLICY\_AUDIT\_EVENT\_TYPE](#Section_04036d49a4824b57995f11d539f2e5b7) enumeration (see section 2.2.4.20). Each element MUST contain one or more of the values in the table below.

If the **MaximumAuditingEventCount** field has a value other than 0, this field MUST NOT be NULL.

| Value | Meaning |
| --- | --- |
| POLICY\_AUDIT\_EVENT\_UNCHANGED  0x00000000 | Leave existing auditing options unchanged for events of this type; used only for set operations. This value cannot be combined with values in this table. |
| POLICY\_AUDIT\_EVENT\_NONE  0x00000004 | Upon updates, this value causes the existing auditing options for events of this type to be deleted and replaced with any other new values specified. If specified by itself, this value cancels all auditing options for events of this type. This value is used only for set operations. |
| POLICY\_AUDIT\_EVENT\_SUCCESS  0x00000001 | When auditing is enabled, audit all successful occurrences of events of the given type. |
| POLICY\_AUDIT\_EVENT\_FAILURE  0x00000002 | When auditing is enabled, audit all unsuccessful occurrences of events of the given type. |

**MaximumAuditEventCount:**  The number of entries in the EventAuditingOptions array.[<17>](#Appendix_A_17" \o "Product behavior note 17)

#### LSAPR\_POLICY\_PRIMARY\_DOM\_INFO

The LSAPR\_POLICY\_PRIMARY\_DOM\_INFO structure defines the server's [**primary domain**](#gt_387021de-3d6b-4372-835f-0d68c50cb496).

The following structure corresponds to the PolicyPrimaryDomainInformation information class.

1. typedef struct \_LSAPR\_POLICY\_PRIMARY\_DOM\_INFO {
2. RPC\_UNICODE\_STRING Name;
3. PRPC\_SID Sid;
4. } LSAPR\_POLICY\_PRIMARY\_DOM\_INFO,
5. \*PLSAPR\_POLICY\_PRIMARY\_DOM\_INFO;

**Name:**  This field contains a name for the primary domain that is subject to the restrictions of a NetBIOS name, as specified in [[RFC1088]](https://go.microsoft.com/fwlink/?LinkId=90266). The value SHOULD be used (by implementations external to this protocol) to identify the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) via the NetBIOS API, as specified in [RFC1088].

**Sid:**  The [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the primary domain.

#### LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO

The LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO structure contains information about the server's [**account domain**](#gt_b56f14e3-d874-48bc-837b-5e812ee1a96e). The following structure corresponds to the PolicyAccountDomainInformation and PolicyLocalAccountDomainInformation information classes.

1. typedef struct \_LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO {
2. RPC\_UNICODE\_STRING DomainName;
3. PRPC\_SID DomainSid;
4. } LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO,
5. \*PLSAPR\_POLICY\_ACCOUNT\_DOM\_INFO;

**DomainName:**  This field contains a name for the account domain that is subjected to the restrictions of a NetBIOS name, as specified in [[RFC1088]](https://go.microsoft.com/fwlink/?LinkId=90266). This value SHOULD be used (by implementations external to this protocol) to identify the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) via the NetBIOS API, as specified in [RFC1088].

**DomainSid:**  The [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the account domain. This field MUST NOT be NULL.

#### LSAPR\_POLICY\_PD\_ACCOUNT\_INFO

The LSAPR\_POLICY\_PD\_ACCOUNT\_INFO structure is obsolete and exists for backward compatibility purposes only.

**Name:** Represents the name of an account in the domain that is to be used for authentication and name/ID lookup requests.

1. typedef struct \_LSAPR\_POLICY\_PD\_ACCOUNT\_INFO {
2. RPC\_UNICODE\_STRING Name;
3. } LSAPR\_POLICY\_PD\_ACCOUNT\_INFO,
4. \*PLSAPR\_POLICY\_PD\_ACCOUNT\_INFO;

#### POLICY\_LSA\_SERVER\_ROLE

The POLICY\_LSA\_SERVER\_ROLE enumeration takes one of two possible values, depending on which capacity the [**account domain**](#gt_b56f14e3-d874-48bc-837b-5e812ee1a96e) database is in: primary or backup. Certain operations of the protocol are allowed only against a primary account database. On non–[**domain controller**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) machines, the account domain database is in primary state. On domain controllers, if the machine is the [**primary domain controller (PDC) role owner**](#gt_e42e52b3-e44f-4284-9c1b-e161f81ea516), then the account domain database is in primary state; otherwise, it is in backup state.

1. typedef enum \_POLICY\_LSA\_SERVER\_ROLE
2. {
3. PolicyServerRoleBackup = 2,
4. PolicyServerRolePrimary
5. } POLICY\_LSA\_SERVER\_ROLE,
6. \*PPOLICY\_LSA\_SERVER\_ROLE;

**PolicyServerRoleBackup:** A backup account database.

**PolicyServerRolePrimary:** A primary account database.

#### POLICY\_LSA\_SERVER\_ROLE\_INFO

The POLICY\_LSA\_SERVER\_ROLE\_INFO structure is used to allow callers to query and set whether the [**account domain**](#gt_b56f14e3-d874-48bc-837b-5e812ee1a96e) database acts as the primary copy or backup copy. The following structure corresponds to the PolicyLsaServerRoleInformation information class.

1. typedef struct \_POLICY\_LSA\_SERVER\_ROLE\_INFO {
2. POLICY\_LSA\_SERVER\_ROLE LsaServerRole;
3. } POLICY\_LSA\_SERVER\_ROLE\_INFO,
4. \*PPOLICY\_LSA\_SERVER\_ROLE\_INFO;

**LsaServerRole:**  One of the values of the [POLICY\_LSA\_SERVER\_ROLE](#Section_620010b4b4394d46893acb67246de5fc) enumeration on return.

#### LSAPR\_POLICY\_REPLICA\_SRCE\_INFO

The LSAPR\_POLICY\_REPLICA\_SRCE\_INFO structure corresponds to the PolicyReplicaSourceInformation information class.

1. typedef struct \_LSAPR\_POLICY\_REPLICA\_SRCE\_INFO {
2. RPC\_UNICODE\_STRING ReplicaSource;
3. RPC\_UNICODE\_STRING ReplicaAccountName;
4. } LSAPR\_POLICY\_REPLICA\_SRCE\_INFO,
5. \*PLSAPR\_POLICY\_REPLICA\_SRCE\_INFO;

**ReplicaSource:**  A string.

**ReplicaAccountName:**  A string.

#### POLICY\_MODIFICATION\_INFO

The POLICY\_MODIFICATION\_INFO structure is obsolete and exists for backward compatibility purposes only. Callers of this protocol MUST NOT be able to set or retrieve this structure.

1. typedef struct \_POLICY\_MODIFICATION\_INFO {
2. LARGE\_INTEGER ModifiedId;
3. LARGE\_INTEGER DatabaseCreationTime;
4. } POLICY\_MODIFICATION\_INFO,
5. \*PPOLICY\_MODIFICATION\_INFO;

**ModifiedId:**  A 64-bit unsigned integer that is incremented each time anything in the Local Security Authority (LSA) database is modified.

**DatabaseCreationTime:**  The date and time when the LSA database was created. It is a 64-bit value that represents the number of 100-nanosecond intervals since January 1, 1601, [**UTC**](#gt_f2369991-a884-4843-a8fa-1505b6d5ece7).

#### POLICY\_AUDIT\_FULL\_SET\_INFO

The POLICY\_AUDIT\_FULL\_SET\_INFO structure contains information to set on the server that is controlling audit log behavior. The following structure corresponds to the PolicyAuditFullSetInformation information class. This information class is not supported.

1. typedef struct \_POLICY\_AUDIT\_FULL\_SET\_INFO {
2. unsigned char ShutDownOnFull;
3. } POLICY\_AUDIT\_FULL\_SET\_INFO,
4. \*PPOLICY\_AUDIT\_FULL\_SET\_INFO;

**ShutDownOnFull:**  A nonzero value means that the system MUST shut down when the event log is full, while zero means that the system MUST NOT shut down when the event log is full.

#### POLICY\_AUDIT\_FULL\_QUERY\_INFO

The POLICY\_AUDIT\_FULL\_QUERY\_INFO structure is used to query information about the state of the audit log on the server. The following structure corresponds to the PolicyAuditFullQueryInformation information class.

This information class is obsolete and exists for backward compatibility purposes only.

1. typedef struct \_POLICY\_AUDIT\_FULL\_QUERY\_INFO {
2. unsigned char ShutDownOnFull;
3. unsigned char LogIsFull;
4. } POLICY\_AUDIT\_FULL\_QUERY\_INFO,
5. \*PPOLICY\_AUDIT\_FULL\_QUERY\_INFO;

**ShutDownOnFull:**  This field indicates whether the system MUST shut down when the event log is full.

**LogIsFull:**  This field indicates whether the event log is full or not.

#### LSAPR\_POLICY\_DNS\_DOMAIN\_INFO

The LSAPR\_POLICY\_DNS\_DOMAIN\_INFO structure is used to allow callers to query and set the server's [**primary domain**](#gt_387021de-3d6b-4372-835f-0d68c50cb496).[<18>](#Appendix_A_18" \o "Product behavior note 18)

The following structure corresponds to the PolicyDnsDomainInformation and PolicyDnsDomainInformationInt information classes.

1. typedef struct \_LSAPR\_POLICY\_DNS\_DOMAIN\_INFO {
2. RPC\_UNICODE\_STRING Name;
3. RPC\_UNICODE\_STRING DnsDomainName;
4. RPC\_UNICODE\_STRING DnsForestName;
5. GUID DomainGuid;
6. PRPC\_SID Sid;
7. } LSAPR\_POLICY\_DNS\_DOMAIN\_INFO,
8. \*PLSAPR\_POLICY\_DNS\_DOMAIN\_INFO;

**Name:**  This field contains a name for the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) that is subject to the restrictions of a NetBIOS name, as specified in [[RFC1088]](https://go.microsoft.com/fwlink/?LinkId=90266). This value SHOULD be used (by implementations external to this protocol) to identify the domain via the NetBIOS API, as specified in [RFC1088].

**DnsDomainName:**  The fully qualified [**DNS name**](#gt_102a36e2-f66f-49e2-bee3-558736b2ecd5) of the domain.

**DnsForestName:**  The fully qualified DNS name of the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) containing this domain.

**DomainGuid:**  The [**globally unique identifier (GUID)**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1), as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.3.4.1, of the domain.

**Sid:**  The [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the domain.

#### POLICY\_DOMAIN\_INFORMATION\_CLASS

The POLICY\_DOMAIN\_INFORMATION\_CLASS enumeration type contains values that specify the type of policy being queried or set by the client.

1. typedef enum \_POLICY\_DOMAIN\_INFORMATION\_CLASS
2. {
3. PolicyDomainQualityOfServiceInformation = 1,
4. PolicyDomainEfsInformation = 2,
5. PolicyDomainKerberosTicketInformation = 3
6. } POLICY\_DOMAIN\_INFORMATION\_CLASS,
7. \*PPOLICY\_DOMAIN\_INFORMATION\_CLASS;

The values in this enumeration are used in defining the contents of the [LSAPR\_POLICY\_DOMAIN\_INFORMATION](#Section_1a9c523ba67a485f8f8b8fca05ca9334) union.

#### LSAPR\_POLICY\_DOMAIN\_INFORMATION

The LSAPR\_POLICY\_DOMAIN\_INFORMATION union is defined as follows, where the structure depends on the [POLICY\_DOMAIN\_INFORMATION\_CLASS](#Section_566a61fc2e9947c899ca62f7e22cb15d) that is specified in the message.

1. typedef
2. [switch\_type(POLICY\_DOMAIN\_INFORMATION\_CLASS)]
3. union \_LSAPR\_POLICY\_DOMAIN\_INFORMATION {
4. [case(PolicyDomainQualityOfServiceInformation)]
5. POLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO PolicyDomainQualityOfServiceInfo;
6. [case(PolicyDomainEfsInformation)]
7. LSAPR\_POLICY\_DOMAIN\_EFS\_INFO PolicyDomainEfsInfo;
8. [case(PolicyDomainKerberosTicketInformation)]
9. POLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO PolicyDomainKerbTicketInfo;
10. } LSAPR\_POLICY\_DOMAIN\_INFORMATION,
11. \*PLSAPR\_POLICY\_DOMAIN\_INFORMATION;

**PolicyDomainQualityOfServiceInfo:**  The complete description is as specified in section [2.2.4.17](#Section_38bd52a04514468fb342d7421a51a316).[<19>](#Appendix_A_19" \o "Product behavior note 19)

**PolicyDomainEfsInfo:**  The complete description is as specified in section [2.2.4.18](#Section_3ba6e751cf914d87a74c488bb927a54c).

**PolicyDomainKerbTicketInfo:**  The complete description is as specified in section [2.2.4.19](#Section_afcc492012d348e0ab95a8989ebbd41d).

#### POLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO

The POLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO structure is obsolete and exists for backward compatibility purposes only.

1. typedef struct \_POLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO {
2. unsigned long QualityOfService;
3. } POLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO,
4. \*PPOLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO;

**QualityOfService:**  Quality of [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) of the responder. MUST be set to zero when sent and MUST be ignored on receipt.

#### LSAPR\_POLICY\_DOMAIN\_EFS\_INFO

The LSAPR\_POLICY\_DOMAIN\_EFS\_INFO structure communicates a counted binary byte array.

1. typedef struct \_LSAPR\_POLICY\_DOMAIN\_EFS\_INFO {
2. unsigned long InfoLength;
3. [size\_is(InfoLength)] unsigned char\* EfsBlob;
4. } LSAPR\_POLICY\_DOMAIN\_EFS\_INFO,
5. \*PLSAPR\_POLICY\_DOMAIN\_EFS\_INFO;

**InfoLength:**  The count of bytes in the **EfsBlob**.

**EfsBlob:**  An array of bytes, of size **InfoLength** bytes. If the value of **InfoLength** is other than 0, this field MUST NOT be NULL. The syntax of this blob SHOULD[<20>](#Appendix_A_20" \o "Product behavior note 20) conform to the layout specified in [[MS-GPEF]](%5bMS-GPEF%5d.pdf#Section_14d3fd83753741a2af398e52c19ef0e3) section 2.2.1.2.1.

#### POLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO

The POLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO structure communicates policy information about the Kerberos security provider.

1. typedef struct \_POLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO {
2. unsigned long AuthenticationOptions;
3. LARGE\_INTEGER MaxServiceTicketAge;
4. LARGE\_INTEGER MaxTicketAge;
5. LARGE\_INTEGER MaxRenewAge;
6. LARGE\_INTEGER MaxClockSkew;
7. LARGE\_INTEGER Reserved;
8. } POLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO,
9. \*PPOLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO;

**AuthenticationOptions:**  Optional flags that affect validations performed during authentication.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | V  C | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Where the bits are defined as:

| Value | Description |
| --- | --- |
| VC  POLICY\_KERBEROS\_VALIDATE\_CLIENT (0x00000080) | This is the only flag that is currently defined. When this bit is set, the AuthenticationOptions flag of the Key Distribution Center (KDC) configuration setting will be set to POLICY\_KERBEROS\_VALIDATE\_CLIENT (as described in [[MS-KILE]](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9) section 3.3.1). All other bits SHOULD be set to 0 and ignored upon receipt. |

**MaxServiceTicketAge:**  This is in units of 10^(-7) seconds. It corresponds to Maximum ticket lifetime (as specified in [[RFC4120]](https://go.microsoft.com/fwlink/?LinkId=90458) section 8.2) for [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) tickets only. The default value of this setting is 10 hours.

**MaxTicketAge:**  This is in units of 10^(-7) seconds. It corresponds to the Maximum ticket lifetime (as specified in [RFC4120] section 8.2) for ticket-granting ticket (TGT) only. The default value of this setting is 10 hours.

**MaxRenewAge:**  This is in units of 10^(-7) seconds. It corresponds to the Maximum renewable lifetime, as specified in [RFC4120] section 8.2. The default value of this setting is one week.

**MaxClockSkew:**  This is in units of 10^(-7) seconds. It corresponds to the Acceptable clock skew, as specified in [RFC4120] section 8.2. The default value of this setting is five minutes.

**Reserved:**  The value of this field SHOULD be set to zero when sent or on receipt.

#### POLICY\_AUDIT\_EVENT\_TYPE

1. typedef enum \_POLICY\_AUDIT\_EVENT\_TYPE
2. {
3. AuditCategorySystem = 0,
4. AuditCategoryLogon,
5. AuditCategoryObjectAccess,
6. AuditCategoryPrivilegeUse,
7. AuditCategoryDetailedTracking,
8. AuditCategoryPolicyChange,
9. AuditCategoryAccountManagement,
10. AuditCategoryDirectoryServiceAccess,
11. AuditCategoryAccountLogon
12. } POLICY\_AUDIT\_EVENT\_TYPE,
13. \*PPOLICY\_AUDIT\_EVENT\_TYPE;

**AuditCategorySystem:** Manages auditing of system-related events

**AuditCategoryLogon:** Manages auditing of account logon events

**AuditCategoryObjectAccess:** Manages auditing of object access events

**AuditCategoryPrivilegeUse:** Manages auditing of [**privilege**](#gt_d8092e10-b227-4b44-b015-511bb8178940) use events

**AuditCategoryDetailedTracking:** Manages detailed auditing

**AuditCategoryPolicyChange:** Manages auditing of policy change events

**AuditCategoryAccountManagement:** Manages auditing of account management events

**AuditCategoryDirectoryServiceAccess:** Manages auditing of [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) access events

**AuditCategoryAccountLogon:** Manages auditing of account logon events

The values in this enumeration are used as indices into the **EventAuditingOptions** field of the [LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO](#Section_d00fc364577d4ed0b3a5952d78b67695) structure (see section 2.2.4.4).

### Account Query/Set Data Types

#### LSAPR\_ACCOUNT\_INFORMATION

The LSAPR\_ACCOUNT\_INFORMATION structure specifies a [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d).

1. typedef struct \_LSAPR\_ACCOUNT\_INFORMATION {
2. PRPC\_SID Sid;
3. } LSAPR\_ACCOUNT\_INFORMATION,
4. \*PLSAPR\_ACCOUNT\_INFORMATION;

**Sid:**  This field contains the SID of the security principal. This field MUST NOT be NULL.

#### LSAPR\_ACCOUNT\_ENUM\_BUFFER

The LSAPR\_ACCOUNT\_ENUM\_BUFFER structure specifies a collection of [**security principal**](#gt_f3ef2572-95cf-4c5c-b3c9-551fd648f409) [**SIDs**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) represented in an array of structures of type LSAPR\_ACCOUNT\_INFORMATION.

1. typedef struct \_LSAPR\_ACCOUNT\_ENUM\_BUFFER {
2. unsigned long EntriesRead;
3. [size\_is(EntriesRead)] PLSAPR\_ACCOUNT\_INFORMATION Information;
4. } LSAPR\_ACCOUNT\_ENUM\_BUFFER,
5. \*PLSAPR\_ACCOUNT\_ENUM\_BUFFER;

**EntriesRead:**  This field contains the number of security principals.

**Information:**  This field contains a set of structures that define the security principal SID, as specified in section [2.2.5.1](#Section_98540c1c09cc4ee2934acdde3de0c77f). If the EntriesRead field has a value other than 0, this field MUST NOT be NULL.

#### LSAPR\_USER\_RIGHT\_SET

The LSAPR\_USER\_RIGHT\_SET structure specifies a collection of user rights.

1. typedef struct \_LSAPR\_USER\_RIGHT\_SET {
2. [range(0,256)] unsigned long Entries;
3. [size\_is(Entries)] PRPC\_UNICODE\_STRING UserRights;
4. } LSAPR\_USER\_RIGHT\_SET,
5. \*PLSAPR\_USER\_RIGHT\_SET;

**Entries:**  This field contains the number of rights.[<21>](#Appendix_A_21" \o "Product behavior note 21)

**UserRights:**  An array of strings specifying the rights. These can be string names corresponding to either [**privilege**](#gt_d8092e10-b227-4b44-b015-511bb8178940) names or system access names, as specified in section [3.1.1.2](#Section_3594f0d3574b4d55a0a435d23885f3ee). If the **Entries** field has a value other than 0, this field MUST NOT be NULL.

#### LSAPR\_LUID\_AND\_ATTRIBUTES

The LSAPR\_LUID\_AND\_ATTRIBUTES structure is a tuple defining a [**locally unique identifier (LUID)**](#gt_96b64af9-1896-4bde-b988-54d469c5affd) and a field defining the attributes of the LUID.

1. typedef struct \_LSAPR\_LUID\_AND\_ATTRIBUTES {
2. LUID Luid;
3. unsigned long Attributes;
4. } LSAPR\_LUID\_AND\_ATTRIBUTES,
5. \*PLSAPR\_LUID\_AND\_ATTRIBUTES;

**Luid:**  The locally unique identifier.

**Attributes:**  This field contains bitmapped values that define the properties of the [**privilege**](#gt_d8092e10-b227-4b44-b015-511bb8178940) set. One or more of the following flags can be set.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | E | D |

**D:** The privilege is enabled by default.

**E:** The privilege is enabled.

All other bits SHOULD be 0 and ignored upon receipt.

#### LSAPR\_PRIVILEGE\_SET

The LSAPR\_PRIVILEGE\_SET structure defines a set of [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) that belong to an account.

1. typedef struct \_LSAPR\_PRIVILEGE\_SET {
2. [range(0,1000)] unsigned long PrivilegeCount;
3. unsigned long Control;
4. [size\_is(PrivilegeCount)] LSAPR\_LUID\_AND\_ATTRIBUTES Privilege[\*];
5. } LSAPR\_PRIVILEGE\_SET,
6. \*PLSAPR\_PRIVILEGE\_SET;

**PrivilegeCount:**  This field contains the number of privileges.[<22>](#Appendix_A_22" \o "Product behavior note 22)

**Control:**  This field contains bitmapped values that define the properties of the privilege set.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | O |

**O:** Valid for a set operation indicating that all specified privileges that are not already assigned are to be assigned.

All other bits SHOULD be set to zero when sent, and ignored on receipt.

**Privilege:**  An array of [LSAPR\_LUID\_AND\_ATTRIBUTES](#Section_03c834c0f3104e0c832eb6e7688364d1) structures. If the **PrivilegeCount** field has a value different than 0, this field MUST NOT be NULL.

### Secret Query/Set Data Types

#### LSAPR\_CR\_CIPHER\_VALUE

The LSAPR\_CR\_CIPHER\_VALUE structure is a counted buffer of bytes containing a [**secret object**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d).

1. typedef struct \_LSAPR\_CR\_CIPHER\_VALUE {
2. [range(0,131088)] unsigned long Length;
3. [range(0,131088)] unsigned long MaximumLength;
4. [size\_is(MaximumLength), length\_is(Length)]
5. unsigned char\* Buffer;
6. } LSAPR\_CR\_CIPHER\_VALUE,
7. \*PLSAPR\_CR\_CIPHER\_VALUE;

**Length:**  This field contains the number of valid bytes in the **Buffer** field.[<23>](#Appendix_A_23" \o "Product behavior note 23)

**MaximumLength:**  This field contains the number of allocated bytes in the **Buffer** field.[<24>](#Appendix_A_24" \o "Product behavior note 24)

**Buffer:**  This field contains the actual secret data. If the value of the **MaximumLength** field is greater than 0, this field MUST contain a non-NULL value. This field is always encrypted using algorithms as specified in section [5.1.2](#Section_cce8ae7801384129954ec65e0c0bffed).

### Trusted Domain Query/Set Data Types

#### LSAPR\_TRUST\_INFORMATION

The LSAPR\_TRUST\_INFORMATION structure identifies a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

1. typedef struct \_LSAPR\_TRUST\_INFORMATION {
2. RPC\_UNICODE\_STRING Name;
3. PRPC\_SID Sid;
4. } LSAPR\_TRUST\_INFORMATION,
5. \*PLSAPR\_TRUST\_INFORMATION;

**Name:**  This field contains a name for the domain that is subject to the restrictions of a NetBIOS name, as specified in [[RFC1088]](https://go.microsoft.com/fwlink/?LinkId=90266). This value SHOULD be used (by implementations external to this protocol) to identify the domain via the NetBIOS, as specified in [RFC1088].

**Sid:**  The [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the domain. This field MUST NOT be NULL.

#### TRUSTED\_INFORMATION\_CLASS

The TRUSTED\_INFORMATION\_CLASS enumeration type contains values that specify the type of [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4) information queried or set by the client.

1. typedef enum \_TRUSTED\_INFORMATION\_CLASS
2. {
3. TrustedDomainNameInformation = 1,
4. TrustedControllersInformation,
5. TrustedPosixOffsetInformation,
6. TrustedPasswordInformation,
7. TrustedDomainInformationBasic,
8. TrustedDomainInformationEx,
9. TrustedDomainAuthInformation,
10. TrustedDomainFullInformation,
11. TrustedDomainAuthInformationInternal,
12. TrustedDomainFullInformationInternal,
13. TrustedDomainInformationEx2Internal,
14. TrustedDomainFullInformation2Internal,
15. TrustedDomainSupportedEncryptionTypes
16. } TRUSTED\_INFORMATION\_CLASS,
17. \*PTRUSTED\_INFORMATION\_CLASS;

**TrustedDomainNameInformation:** The trusted domain information contains the LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO structure specified in section [2.2.7.4](#Section_71C5724F447E452C9CB9A0FD90D88594).

**TrustedControllersInformation:** The trusted domain information contains the LSAPR\_TRUSTED\_CONTROLLERS\_INFO structure specified in section [2.2.7.5](#Section_5382BD8969C646F2BEB17B70E5BEFBC5).

**TrustedPosixOffsetInformation:** The trusted domain information contains the TRUSTED\_POSIX\_OFFSET\_INFO structure specified in section [2.2.7.6](#Section_B091EE7EF5C34B4885671B08EA002221).

**TrustedPasswordInformation:** The trusted domain information contains the LSAPR\_TRUSTED\_PASSWORD\_INFO structure specified in section [2.2.7.7](#Section_33D7A9E4C9CA40219627337D89E656A3).

**TrustedDomainInformationBasic:** The trusted domain information contains the LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_BASIC structure specified in section [2.2.7.8](#Section_C101591DE1B042BD8CC5F8866C3B5757).

**TrustedDomainInformationEx:** The trusted domain information contains the LSAPR\_TRUSTED\_ DOMAIN\_INFORMATION\_EX structure specified in section [2.2.7.9](#Section_F28F42B7173C4CDA98093FE4A5213AB3).

**TrustedDomainAuthInformation:** The trusted domain information contains the LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION structure specified in section [2.2.7.11](#Section_084FDB6B5BC349129AED0257159996DD).

**TrustedDomainFullInformation:** The trusted domain information contains the LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION structure specified in section [2.2.7.13](#Section_9F9FEEBCE9E141C18C4802F83A227A14).

**TrustedDomainAuthInformationInternal:** The trusted domain information contains the LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL structure specified in section [2.2.7.12](#Section_3B1C61FE6F074D83AF543A381DE5C5D1).

**TrustedDomainFullInformationInternal:** The trusted domain information contains the LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL structure specified in section [2.2.7.14](#Section_2E9E2C847B004FB18DE588D4CFEDD2B3).

**TrustedDomainInformationEx2Internal:** The trusted domain information contains the LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 structure specified in section [2.2.7.10](#Section_DD92D4D9227F4EF1B42BEF3F056F8AAA).

**TrustedDomainFullInformation2Internal:** The trusted domain information contains the LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2 structure specified in section [2.2.7.15](#Section_E529D0945DE44738ADC4EFA1A7D1106F).

**TrustedDomainSupportedEncryptionTypes:** The trusted domain information contains the TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES structure specified in section [2.2.7.18](#Section_7C519A643DC14BE6A17D76817CFF6E39).

The following citation contains a timeline of when each enumeration value was introduced.[<25>](#Appendix_A_25" \o "Product behavior note 25)

The values in this enumeration are used in defining the contents of the [LSAPR\_TRUSTED\_DOMAIN\_INFO](#Section_65564571dd0d49a98a2a6dba8ab57091) union.

#### LSAPR\_TRUSTED\_DOMAIN\_INFO

The LSAPR\_TRUSTED\_DOMAIN\_INFO union is defined as follows, where the structure depends on the [TRUSTED\_INFORMATION\_CLASS](#Section_360691136c3845e8920e17f8ef36f578) that is specified in the message.

1. typedef
2. [switch\_type(TRUSTED\_INFORMATION\_CLASS)]
3. union \_LSAPR\_TRUSTED\_DOMAIN\_INFO {
4. [case(TrustedDomainNameInformation)]
5. LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO TrustedDomainNameInfo;
6. [case(TrustedControllersInformation)]
7. LSAPR\_TRUSTED\_CONTROLLERS\_INFO TrustedControllersInfo;
8. [case(TrustedPosixOffsetInformation)]
9. TRUSTED\_POSIX\_OFFSET\_INFO TrustedPosixOffsetInfo;
10. [case(TrustedPasswordInformation)]
11. LSAPR\_TRUSTED\_PASSWORD\_INFO TrustedPasswordInfo;
12. [case(TrustedDomainInformationBasic)]
13. LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_BASIC TrustedDomainInfoBasic;
14. [case(TrustedDomainInformationEx)]
15. LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX TrustedDomainInfoEx;
16. [case(TrustedDomainAuthInformation)]
17. LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION TrustedAuthInfo;
18. [case(TrustedDomainFullInformation)]
19. LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION TrustedFullInfo;
20. [case(TrustedDomainAuthInformationInternal)]
21. LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL TrustedAuthInfoInternal;
22. [case(TrustedDomainFullInformationInternal)]
23. LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL TrustedFullInfoInternal;
24. [case(TrustedDomainInformationEx2Internal)]
25. LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 TrustedDomainInfoEx2;
26. [case(TrustedDomainFullInformation2Internal)]
27. LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2 TrustedFullInfo2;
28. [case(TrustedDomainSupportedEncryptionTypes)]
29. TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES TrustedDomainSETs;
30. } LSAPR\_TRUSTED\_DOMAIN\_INFO,
31. \*PLSAPR\_TRUSTED\_DOMAIN\_INFO;

**TrustedDomainNameInfo:**  An instance of the LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO structure specified in section [2.2.7.4](#Section_71c5724f447e452c9cb9a0fd90d88594).

**TrustedControllersInfo:**  An instance of the LSAPR\_TRUSTED\_CONTROLLERS\_INFO structure specified in section [2.2.7.5](#Section_5382bd8969c646f2beb17b70e5befbc5).

**TrustedPosixOffsetInfo:**  An instance of the TRUSTED\_POSIX\_OFFSET\_INFO structure specified in section [2.2.7.6](#Section_b091ee7ef5c34b4885671b08ea002221).

**TrustedPasswordInfo:**  An instance of the LSAPR\_TRUSTED\_PASSWORD\_INFO structure specified in section [2.2.7.7](#Section_33d7a9e4c9ca40219627337d89e656a3).

**TrustedDomainInfoBasic:**  An instance of the LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_BASIC structure specified in section [2.2.7.8](#Section_c101591de1b042bd8cc5f8866c3b5757).

**TrustedDomainInfoEx:**  An instance of the LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX structure specified in section [2.2.7.9](#Section_f28f42b7173c4cda98093fe4a5213ab3).

**TrustedAuthInfo:**  An instance of the LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION structure specified in section [2.2.7.11](#Section_084fdb6b5bc349129aed0257159996dd).

**TrustedFullInfo:**  An instance of the LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION structure specified in section [2.2.7.13](#Section_9f9feebce9e141c18c4802f83a227a14).

**TrustedAuthInfoInternal:**  An instance of the LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL structure specified in section [2.2.7.12](#Section_3b1c61fe6f074d83af543a381de5c5d1).

**TrustedFullInfoInternal:**  An instance of the LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL structure specified in section [2.2.7.14](#Section_2e9e2c847b004fb18de588d4cfedd2b3).

**TrustedDomainInfoEx2:**  An instance of the LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 structure specified in section [2.2.7.10](#Section_dd92d4d9227f4ef1b42bef3f056f8aaa).

**TrustedFullInfo2:**  An instance of the LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2 structure specified in section [2.2.7.15](#Section_e529d0945de44738adc4efa1a7d1106f).

**TrustedDomainSETs:**  An instance of the TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES structure specified in section [2.2.7.18](#Section_7c519a643dc14be6a17d76817cff6e39).

#### LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO

The LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO structure is used to communicate the name of a [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4). The following structure corresponds to the TrustedDomainNameInformation information class.

1. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO {
2. RPC\_UNICODE\_STRING Name;
3. } LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO,
4. \*PLSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO;

**Name:**  This field contains a name for the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) that is subject to the restrictions of a NetBIOS name, as specified in [[RFC1088]](https://go.microsoft.com/fwlink/?LinkId=90266). This field SHOULD be used (by implementations external to this protocol) to identify the domain via the NetBIOS API, as specified in [RFC1088].

#### LSAPR\_TRUSTED\_CONTROLLERS\_INFO

The LSAPR\_TRUSTED\_CONTROLLERS\_INFO structure is used to communicate a set of names of [**domain controllers (DCs)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) in a [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4). The following structure corresponds to the TrustedControllersInformation information class.

1. typedef struct \_LSAPR\_TRUSTED\_CONTROLLERS\_INFO {
2. [range(0,5)] unsigned long Entries;
3. [size\_is(Entries)] PRPC\_UNICODE\_STRING Names;
4. } LSAPR\_TRUSTED\_CONTROLLERS\_INFO,
5. \*PLSAPR\_TRUSTED\_CONTROLLERS\_INFO;

**Entries:**  The count of names.[<26>](#Appendix_A_26" \o "Product behavior note 26)

**Names:**  This field contains an array of DC names that are subject to the restrictions of a NetBIOS name, as specified in [[RFC1088]](https://go.microsoft.com/fwlink/?LinkId=90266). This field SHOULD be used (by implementations external to this protocol) to identify the DCs via the NetBIOS API, as specified in [RFC1088]. If the **Entries** field has a value other than 0, this field MUST NOT be NULL.

#### TRUSTED\_POSIX\_OFFSET\_INFO

The TRUSTED\_POSIX\_OFFSET\_INFO structure communicates any offset necessary for POSIX compliance. The following structure corresponds to the TrustedPosixOffsetInformation information class.

1. typedef struct \_TRUSTED\_POSIX\_OFFSET\_INFO {
2. unsigned long Offset;
3. } TRUSTED\_POSIX\_OFFSET\_INFO,
4. \*PTRUSTED\_POSIX\_OFFSET\_INFO;

**Offset:**  The offset to use for the generation of POSIX IDs for users and groups, as specified in "trustPosixOffset" in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6.7.14.

#### LSAPR\_TRUSTED\_PASSWORD\_INFO

The LSAPR\_TRUSTED\_PASSWORD\_INFO structure is used to communicate [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6)-authentication material. The following structure corresponds to the TrustedPasswordInformation information class.

1. typedef struct \_LSAPR\_TRUSTED\_PASSWORD\_INFO {
2. PLSAPR\_CR\_CIPHER\_VALUE Password;
3. PLSAPR\_CR\_CIPHER\_VALUE OldPassword;
4. } LSAPR\_TRUSTED\_PASSWORD\_INFO,
5. \*PLSAPR\_TRUSTED\_PASSWORD\_INFO;

**Password:**  The current authentication material. See section [2.2.6.1](#Section_782eda77b82e413487c9eb5e67f18f06).

**OldPassword:**  The version prior to the current version of the authentication material. See section 2.2.6.1.

#### LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_BASIC

The LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_BASIC type is identical to the [LSAPR\_TRUST\_INFORMATION](#Section_71e86cddae194a029179a2a103b383a0) structure. This type corresponds to the TrustedDomainInformationBasic information class.

This type is declared as follows:

1. typedef LSAPR\_TRUST\_INFORMATION LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_BASIC;

#### LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX

The LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX structure communicates properties of a [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4). The following structure corresponds to the TrustedDomainInformationEx information class. [**Domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) [**trusts**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) are specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6.

1. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX {
2. RPC\_UNICODE\_STRING Name;
3. RPC\_UNICODE\_STRING FlatName;
4. PRPC\_SID Sid;
5. unsigned long TrustDirection;
6. unsigned long TrustType;
7. unsigned long TrustAttributes;
8. } LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX,
9. \*PLSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX;

**Name:**  The [**DNS name**](#gt_102a36e2-f66f-49e2-bee3-558736b2ecd5) of the domain. Maps to the **Name** field, as specified in section [3.1.1.5](#Section_0228f75e9725479cb4cd1cacd667343c).

**FlatName:**  The NetBIOS name of the trusted domain, as specified in [[RFC1088]](https://go.microsoft.com/fwlink/?LinkId=90266). Maps to the **Flat Name** field, as specified in section 3.1.1.5.

**Sid:**  The domain [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d). Maps to the **Security Identifier** field, as specified in section 3.1.1.5.

**TrustDirection:**  This field contains bitmapped values that define the properties of the direction of trust between the local domain and the named domain. One or more of the valid flags can be set. If all bits are 0, the trust is said to be disabled.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | I | O |

**I:** The trust is inbound.

**O:** The trust is outbound.

All other bits SHOULD be 0 and ignored upon receipt.

Maps to the **Trust Direction** field, as specified in section 3.1.1.5.

**TrustType:**  This field specifies the type of trust between the local domain and the named domain.

| Value | Meaning |
| --- | --- |
| 0x00000001 | Trust with a Windows domain that is not running [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90). |
| 0x00000002 | Trust with a Windows domain that is running Active Directory. |
| 0x00000003 | Trust with a non–Windows-compliant Kerberos distribution, as specified in [[RFC4120]](https://go.microsoft.com/fwlink/?LinkId=90458). |
| 0x00000004 | Trust with a distributed computing environment (DCE) realm. This is a historical reference and is not used. |

**Note**  Other values SHOULD NOT be set.

Maps to the **Trust Type** field, as specified in section 3.1.1.5.

**TrustAttributes:**  This field contains bitmapped values that define the attributes of the trust.[<27>](#Appendix_A_27" \o "Product behavior note 27)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| R | R | R | R | R | R | R | R | O | O | R | R | R | R | R | R | R | R | R | R | R | T A P T | T A N C | R | T A R C | T A T E | T A W F | T A C O | T A F T | T A Q D | T A U O | T A N T |

TrustAttribute values are described in section 3.1.1.5. The following table shows how these values map to the Trust Attributes field in section 3.1.1.5.

| Value | Mapping |
| --- | --- |
| TANT (TRUST\_ATTRIBUTE\_NON\_TRANSITIVE) | Trust Attributes: Non-transitive |
| TAUO (TRUST\_ATTRIBUTE\_UPLEVEL\_ONLY) | Trust Attributes: Uplevel only |
| TAQD (TRUST\_ATTRIBUTE\_QUARANTINED\_DOMAIN) | Trust Attributes: Quarantined |
| TAFT (TRUST\_ATTRIBUTE\_FOREST\_TRANSITIVE) | Trust Attributes: Forest trust |
| TACO (TRUST\_ATTRIBUTE\_CROSS\_ORGANIZATION) | Trust Attributes: Cross organization |
| TAWF (TRUST\_ATTRIBUTE\_WITHIN\_FOREST) | Trust Attributes: Within [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) |
| TATE (TRUST\_ATTRIBUTE\_TREAT\_AS\_EXTERNAL) | Trust Attributes: Treat as external |
| TARC (TRUST\_ATTRIBUTE\_USES\_RC4\_ENCRYPTION) | Trust Attributes: Use [**RC4**](#gt_d57eac33-f561-4a08-b148-dfcf29cfb4d8) Encryption (for more information about RC4, see [[SCHNEIER]](https://go.microsoft.com/fwlink/?LinkId=817338) section 17.1). |
| TANC (TRUST\_ATTRIBUTE\_CROSS\_ORGANIZATION\_NO\_TGT\_DELEGATION) | Trust Attributes: Tokens must not be trusted for delegation. |
| TAPT (TRUST\_ATTRIBUTE\_PIM\_TRUST) | Trust Attributes: PrivilegedIdentityManagement (PIM) trust. |
| O | Obsolete. SHOULD be set to 0. |
| R | Reserved for future use. SHOULD be set to zero. |

#### LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2

The LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 structure communicates properties of a [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4). The following structure corresponds to the TrustedDomainInformationEx2Internal information class. [**Domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) [**trusts**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) are specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6.

1. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 {
2. RPC\_UNICODE\_STRING Name;
3. RPC\_UNICODE\_STRING FlatName;
4. PRPC\_SID Sid;
5. unsigned long TrustDirection;
6. unsigned long TrustType;
7. unsigned long TrustAttributes;
8. unsigned long ForestTrustLength;
9. [size\_is(ForestTrustLength)] unsigned char\* ForestTrustInfo;
10. } LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2,
11. \*PLSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2;

**Name:**  The [**DNS name**](#gt_102a36e2-f66f-49e2-bee3-558736b2ecd5) of the domain. Maps to the **Name** field, as specified in section [3.1.1.5](#Section_0228f75e9725479cb4cd1cacd667343c).

**FlatName:**  The NetBIOS name of the trusted domain, as specified in [[RFC1088]](https://go.microsoft.com/fwlink/?LinkId=90266). Maps to the **Flat Name** field, as specified in section 3.1.1.5.

**Sid:**  The domain [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d). Maps to the **Security Identifier** field, as specified in section 3.1.1.5.

**TrustDirection:**  This field contains bitmapped values that define the properties of the direction of trust between the local domain and the named domain. See section [2.2.7.9](#Section_f28f42b7173c4cda98093fe4a5213ab3) for valid values and a description of each flag. Maps to the Trusted Direction field, as specified in section 3.1.1.5.

**TrustType:**  This field specifies the type of trust between the local domain and the named domain. See section 2.2.7.9 for valid values and a description of each value. Maps to the Trusted Type field, as specified in section 3.1.1.5.

**TrustAttributes:**  This field contains bitmapped values that define the attributes of the trust. See section 2.2.7.9 for valid values and a description of each flag. Maps to the Trusted Attributes field, as specified in section 3.1.1.5.

**ForestTrustLength:**  The count of bytes in **ForestTrustInfo**.

**ForestTrustInfo:**  Binary data for the [**forest trust**](#gt_035d9ce5-f117-4251-8d4d-127c462ec4a0). For more information, see "Trust Objects" in [MS-ADTS] section 6.1.6. Maps to the Forest Trust Information field, as specified in section 3.1.1.5. Conversion from this binary format to the [LSA\_FOREST\_TRUST\_INFORMATION](#Section_2993ffabc0c846439a794ff7d31922dc) format is specified in [MS-ADTS] section 6.1.6.9.3. If the **ForestTrustLength** field has a value other than 0, this field MUST NOT be NULL.

#### LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION

The LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION structure communicates authentication material. The following structure corresponds to the TrustedDomainAuthInformation information class. [**Domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) authentication is specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6.9.1. This structure maps to the Incoming and Outgoing Trust Password fields, as specified in section [3.1.1.5](#Section_0228f75e9725479cb4cd1cacd667343c).

1. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION {
2. [range(0,1)] unsigned long IncomingAuthInfos;
3. PLSAPR\_AUTH\_INFORMATION IncomingAuthenticationInformation;
4. PLSAPR\_AUTH\_INFORMATION IncomingPreviousAuthenticationInformation;
5. [range(0,1)] unsigned long OutgoingAuthInfos;
6. PLSAPR\_AUTH\_INFORMATION OutgoingAuthenticationInformation;
7. PLSAPR\_AUTH\_INFORMATION OutgoingPreviousAuthenticationInformation;
8. } LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION,
9. \*PLSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION;

**IncomingAuthInfos:**  The count of [LSAPR\_AUTH\_INFORMATION](#Section_cedb0d1bc7c0448099fc279b06f22a0c) entries (section 2.2.7.17) in the **IncomingAuthenticationInformation** field.[<28>](#Appendix_A_28" \o "Product behavior note 28)

**IncomingAuthenticationInformation:**  An array of LSAPR\_AUTH\_INFORMATION structures. The values are used to compute keys used in inbound trust validation, as specified in [MS-ADTS] section 6.1.6.9.1.

**IncomingPreviousAuthenticationInformation:**  Same as **IncomingAuthenticationInformation**, but the data is the previous version of the authentication information.

**OutgoingAuthInfos:**  The count of LSAPR\_AUTH\_INFORMATION entries in the OutgoingAuthenticationInformation field. [<29>](#Appendix_A_29" \o "Product behavior note 29)

**OutgoingAuthenticationInformation:**  An array of LSAPR\_AUTH\_INFORMATION structures. The values are used to compute keys used in outbound trust validation, as specified in [MS-ADTS] section 6.1.6.9.1.

**OutgoingPreviousAuthenticationInformation:**  Same as **OutgoingAuthenticationInformation**, but the data is the previous version of the authentication information.

#### LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL

The LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL structure communicates authentication material. The following structure corresponds to the TrustedDomainAuthInformationInternal information class. For more information about [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) authentication material, see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6.9.1.

1. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL {
2. LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB AuthBlob;
3. } LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL,
4. \*PLSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL;

**AuthBlob:**  An [LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB](#Section_da8f32a10a164194810d06cc0698595a).

#### LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION

The LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION structure communicates identification, POSIX compatibility, and authentication information for a [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4). The following structure corresponds to the TrustedDomainFullInformation information class.

1. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION {
2. LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX Information;
3. TRUSTED\_POSIX\_OFFSET\_INFO PosixOffset;
4. LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION AuthInformation;
5. } LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION,
6. \*PLSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION;

**Information:**  A structure containing name, [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d), and [**trust attributes**](#gt_0b8230da-0fd8-492c-b84a-d8467c3dc1ef), as specified in section [2.2.7.9](#Section_f28f42b7173c4cda98093fe4a5213ab3).

**PosixOffset:**  Any offset required for POSIX compliance, as specified in section [2.2.7.6](#Section_b091ee7ef5c34b4885671b08ea002221).

**AuthInformation:**  Authentication material, as specified in section [2.2.7.11](#Section_084fdb6b5bc349129aed0257159996dd).

#### LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL

The LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL structure communicates identification and authentication information for a [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4). The following structure corresponds to the TrustedDomainFullInformationInternal information class.

1. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL {
2. LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX Information;
3. TRUSTED\_POSIX\_OFFSET\_INFO PosixOffset;
4. LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL AuthInformation;
5. } LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL,
6. \*PLSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL;

**Information:**  A structure containing name, [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d), and [**trust attributes**](#gt_0b8230da-0fd8-492c-b84a-d8467c3dc1ef), as specified in section [2.2.7.9](#Section_f28f42b7173c4cda98093fe4a5213ab3).

**PosixOffset:**  Any offset required for POSIX compliance, as specified in section [2.2.7.6](#Section_b091ee7ef5c34b4885671b08ea002221).

**AuthInformation:**  Authentication material, as specified in section [2.2.7.12](#Section_3b1c61fe6f074d83af543a381de5c5d1).

#### LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2

The LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2 structure is used to communicate identification, POSIX compatibility, and authentication information for a [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4). The following structure corresponds to the TrustedDomainFullInformation2Internal information class.

1. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2 {
2. LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 Information;
3. TRUSTED\_POSIX\_OFFSET\_INFO PosixOffset;
4. LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION AuthInformation;
5. } LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2,
6. \*PLSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2;

**Information:**  A structure containing name, [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d), and [**trust attributes**](#gt_0b8230da-0fd8-492c-b84a-d8467c3dc1ef), as specified in section [2.2.7.10](#Section_dd92d4d9227f4ef1b42bef3f056f8aaa).

**PosixOffset:**  Any offset required for POSIX compliance, as specified in section [2.2.7.6](#Section_b091ee7ef5c34b4885671b08ea002221).

**AuthInformation:**  Authentication material, as specified in section [2.2.7.11](#Section_084fdb6b5bc349129aed0257159996dd).

#### LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB

The LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB structure contains a counted buffer of authentication material. [**Domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) authentication is specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6.9.1.

1. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB {
2. [range(0,65536)] unsigned long AuthSize;
3. [size\_is(AuthSize)] unsigned char\* AuthBlob;
4. } LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB,
5. \*PLSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB;

**AuthSize:**  The count of bytes in **AuthBlob**.[<30>](#Appendix_A_30" \o "Product behavior note 30)

**AuthBlob:**  An array of bytes containing the authentication material. If the **AuthSize** field has a value other than 0, this field MUST NOT be NULL. Always encrypted using algorithms, as specified in section [5.1.1](#Section_1f5bd3edcfdd42aba2acf0786082bb21). The plaintext layout is in the following format.

The incoming and outgoing authentication information buffer size included at the end of the LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB can be used to extract the incoming and outgoing authentication information buffers from the LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB. Each of these buffers contains the byte offset to both the current and the previous authentication information. This information can be used to extract current and (if any) previous authentication information.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| 512 bytes of random data (512 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CountOutgoingAuthInfos | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ByteOffsetCurrentOutgoingAuthInfo | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ByteOffsetPreviousOutgoingAuthInfo | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CurrentOutgoingAuthInfos (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PreviousOutgoingAuthInfos (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CountIncomingAuthInfos | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ByteOffsetCurrentIncomingAuthInfo | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ByteOffsetPreviousIncomingAuthInfo | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CurrentIncomingAuthInfos (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PreviousIncomingAuthInfos (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| OutgoingAuthInfoSize | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| IncomingAuthInfoSize | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**CountOutgoingAuthInfos (4 bytes):** Specifies the count of entries present in the CurrentOutgoingAuthInfos field. If optional field PreviousOutgoingAuthInfos is present, the number of entries in PreviousOutgoingAuthInfos is also equal to CountOutgoingAuthInfos.

**ByteOffsetCurrentOutgoingAuthInfo (4 bytes):** Specifies the byte offset from the beginning of CountOutgoingAuthInfos to the start of the CurrentOutgoingAuthInfos field. If CountOutgoingAuthInfos is 0, this field MUST be ignored.

**ByteOffsetPreviousOutgoingAuthInfo (4 bytes):** Specifies the byte offset from the beginning of CountOutgoingAuthInfos to the start of the PreviousOutgoingAuthInfos field. If the difference between ByteOffsetPreviousOutgoingAuthInfo and OutgoingAuthInfoSize is 0, the PreviousOutgoingAuthInfos field has zero entries.

**CurrentOutgoingAuthInfo**s: Contains an array of CountOutgoingAuthInfos of [LSAPR\_AUTH\_INFORMATION (section 2.2.7.17)](#Section_cedb0d1bc7c0448099fc279b06f22a0c) entries in self-relative format. Each LSAPR\_AUTH\_INFORMATION entry in the array MUST be 4-byte aligned. When it is necessary to insert unused padding bytes into a buffer for data alignment, such bytes MUST be set to 0.

**PreviousOutgoingAuthInfos**: Contains an array of **CountOutgoingAuthInfos** LSAPR\_AUTH\_INFORMATION entries in self-relative format. See the comments for the **ByteOffsetPreviousOutgoingAuthInfo** field to determine when this field is present. Each LSAPR\_AUTH\_INFORMATION entry in the array MUST be 4-byte aligned. When it is necessary to insert unused padding bytes into a buffer for data alignment, such bytes MUST be set to 0.

**CountIncomingAuthInfos (4 bytes)**: Specifies the count of entries present in the **CurrentIncomingAuthInfos** field. If optional field **PreviousIncomingAuthInfos** is present, the number of entries in **PreviousIncomingAuthInfos** is also equal to **CountIncomingAuthInfos**.

**ByteOffsetCurrentIncomingAuthInfo (4 bytes)**: Specifies the byte offset from the beginning of **CountIncomingAuthInfos** to the start of the **CurrentIncomingAuthInfos** field. If **CountIncomingAuthInfos** is 0, this field MUST be ignored.

**ByteOffsetPreviousIncomingAuthInfo (4 bytes)**: Specifies the byte offset from the beginning of **CountIncomingAuthInfos** to the start of the **PreviousIncomingAuthInfos** field. If the difference between **ByteOffsetPreviousIncomingAuthInfo** and **IncomingAuthInfoSize** is 0, the **PreviousIncomingAuthInfos** field has zero entries.

**CurrentIncomingAuthInfos**: Contains an array of **CountIncomingAuthInfos** LSAPR\_AUTH\_INFORMATION entries in self-relative format. Each LSAPR\_AUTH\_INFORMATION entry in the array MUST be 4-byte aligned. When it is necessary to insert unused padding bytes into a buffer for data alignment, such bytes MUST be set to 0.

**PreviousIncomingAuthInfos**: Contains an array of **CountIncomingAuthInfos** LSAPR\_AUTH\_INFORMATION entries in self-relative format. See the comments for the **ByteOffsetPreviousIncomingAuthInfo** field to determine when this field is present. Each LSAPR\_AUTH\_INFORMATION entry in the array MUST be 4-byte aligned. When it is necessary to insert unused padding bytes into a buffer for data alignment, such bytes MUST be set to 0.

**OutgoingAuthInfoSize (4 bytes)**: Specifies the size, in bytes, of the subportion of the structure from the beginning of the **CountOutgoingAuthInfos** field through the end of the of the **PreviousOutgoingAuthInfos** field.

**IncomingAuthInfoSize (4 bytes)**: Specifies the size, in bytes, of the sub-portion of the structure from the beginning of the **CountIncomingAuthInfos** field through the end of the of the **PreviousIncomingAuthInfos** field.

#### LSAPR\_AUTH\_INFORMATION

The LSAPR\_AUTH\_INFORMATION structure communicates information about authentication between [**trusted domains**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4). [**Domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) authentication is specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6.9.1.

1. typedef struct \_LSAPR\_AUTH\_INFORMATION {
2. LARGE\_INTEGER LastUpdateTime;
3. unsigned long AuthType;
4. [range(0,65536)] unsigned long AuthInfoLength;
5. [size\_is(AuthInfoLength)] unsigned char\* AuthInfo;
6. } LSAPR\_AUTH\_INFORMATION,
7. \*PLSAPR\_AUTH\_INFORMATION;

**LastUpdateTime:**  The date and time when this authentication information was last updated. It is a 64-bit value that represents the number of 100-nanosecond intervals since January 1, 1601, [**UTC**](#gt_f2369991-a884-4843-a8fa-1505b6d5ece7).

**AuthType:**  A type for the AuthInfo, as specified in the following table.

| Value | Meaning |
| --- | --- |
| 0x00000000 | This type MUST be ignored. |
| 0x00000001 | Derived RC4HMAC key. For more information, see [[RFC4757]](https://go.microsoft.com/fwlink/?LinkId=90488). |
| 0x00000002 | A plaintext password. Indicates that the information stored in the attribute is a Unicode plaintext password. If this AuthType is present, Kerberos can then use this password to derive additional key types that are needed to encrypt and decrypt cross-realm TGTs. |
| 0x00000003 | A plaintext password version number that is a single, unsigned long integer consisting of 32 bits. |

**AuthInfoLength:**  The count of bytes in AuthInfo buffer.[<31>](#Appendix_A_31" \o "Product behavior note 31)

**AuthInfo:**  Authentication data that depends on the **AuthType**.

The self-relative form of the LSAPR\_AUTH\_INFORMATION structure is used in [LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB](#Section_da8f32a10a164194810d06cc0698595a); in that case, the structure memory layout looks like the following.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| LastUpdateTime | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AuthType | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AuthInfoLength | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AuthInfo [1 ... AuthInfoLength] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

#### TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES

The TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES structure is used to present the encryption types that are allowed through a [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6).

1. typedef struct \_TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES {
2. unsigned long SupportedEncryptionTypes;
3. } TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES,
4. \*PTRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES;

**SupportedEncryptionTypes:**  This field contains bitmapped values that define the encryption types supported by this trust relationship. The flags can be set in any combination.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | S | A | R | M | C |

**C:** Supports CRC32, as specified in [[RFC3961]](https://go.microsoft.com/fwlink/?LinkId=90450) page 31.

**M:** Supports RSA-MD5, as specified in [RFC3961] page 31.

**R:** Supports RC4-HMAC-MD5, as specified in [[RFC4757]](https://go.microsoft.com/fwlink/?LinkId=90488).

**A:** Supports HMAC-SHA1-96-AES128, as specified in [RFC3961] page 31.

**S:** Supports HMAC-SHA1-96-AES256, as specified in [RFC3961] page 31.

All other bits SHOULD be 0 and ignored upon receipt.

#### LSAPR\_TRUSTED\_ENUM\_BUFFER

The LSAPR\_TRUSTED\_ENUM\_BUFFER structure specifies a collection of [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) information structures of type [LSAPR\_TRUST\_INFORMATION](#Section_71e86cddae194a029179a2a103b383a0).

1. typedef struct \_LSAPR\_TRUSTED\_ENUM\_BUFFER {
2. unsigned long EntriesRead;
3. [size\_is(EntriesRead)] PLSAPR\_TRUST\_INFORMATION Information;
4. } LSAPR\_TRUSTED\_ENUM\_BUFFER,
5. \*PLSAPR\_TRUSTED\_ENUM\_BUFFER;

**EntriesRead:**  This field contains the number of trust information structures.

**Information:**  This field contains a set of structures that define the trust information, as specified in section 2.2.7.1. If the **EntriesRead** field has a value other than 0, this field MUST NOT be NULL.

#### LSAPR\_TRUSTED\_ENUM\_BUFFER\_EX

The LSAPR\_TRUSTED\_ENUM\_BUFFER\_EX structure specifies a collection of [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) information structures of type [LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX](#Section_f28f42b7173c4cda98093fe4a5213ab3).

1. typedef struct \_LSAPR\_TRUSTED\_ENUM\_BUFFER\_EX {
2. unsigned long EntriesRead;
3. [size\_is(EntriesRead)] PLSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX EnumerationBuffer;
4. } LSAPR\_TRUSTED\_ENUM\_BUFFER\_EX,
5. \*PLSAPR\_TRUSTED\_ENUM\_BUFFER\_EX;

**EntriesRead:**  This field contains the number of trust information structures.

**EnumerationBuffer:**  This field contains a set of structures that define the trust information, as specified in section 2.2.7.9. If the **EntriesRead** field has a value other than 0, this field MUST NOT be NULL.

#### LSA\_FOREST\_TRUST\_RECORD

The LSA\_FOREST\_TRUST\_RECORD structure is used to communicate the type, creation time, and data for a [**forest trust**](#gt_035d9ce5-f117-4251-8d4d-127c462ec4a0) record. The data is determined by the [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) type as follows in the definition of the contained union.

1. typedef struct \_LSA\_FOREST\_TRUST\_RECORD {
2. unsigned long Flags;
3. LSA\_FOREST\_TRUST\_RECORD\_TYPE ForestTrustType;
4. LARGE\_INTEGER Time;
5. [switch\_type(LSA\_FOREST\_TRUST\_RECORD\_TYPE), switch\_is(ForestTrustType)]
6. union {
7. [case(ForestTrustTopLevelName,ForestTrustTopLevelNameEx)]
8. LSA\_UNICODE\_STRING TopLevelName;
9. [case(ForestTrustDomainInfo)]
10. LSA\_FOREST\_TRUST\_DOMAIN\_INFO DomainInfo;
11. [default] LSA\_FOREST\_TRUST\_BINARY\_DATA Data;
12. } ForestTrustData;
13. } LSA\_FOREST\_TRUST\_RECORD,
14. \*PLSA\_FOREST\_TRUST\_RECORD;

**Flags:**  The following table specifies the possible flags.

**Note**  Some flag values are reused for different [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) record types. See the Meaning column for more information.

| Value | Meaning |
| --- | --- |
| LSA\_TLN\_DISABLED\_NEW  0x00000001 | The top-level name trust record is disabled during initial creation.  **Note**  This flag MUST be used only with forest trust record types of ForestTrustTopLevelName and ForestTrustTopLevelNameEx. |
| LSA\_TLN\_DISABLED\_ADMIN  0x00000002 | The top-level name trust record is disabled by the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) administrator.  **Note**  This flag MUST be used only with forest trust record types of ForestTrustTopLevelName and ForestTrustTopLevelNameEx. |
| LSA\_TLN\_DISABLED\_CONFLICT  0x00000004 | The top-level name trust record is disabled due to a conflict.  **Note**  This flag MUST be used only with forest trust record types of ForestTrustTopLevelName and ForestTrustTopLevelNameEx. |
| LSA\_SID\_DISABLED\_ADMIN  0x00000001 | The domain information trust record is disabled by the domain administrator.  **Note**  This flag MUST be used only with a forest trust record type of ForestTrustDomainInfo. |
| LSA\_SID\_DISABLED\_CONFLICT  0x00000002 | The domain information trust record is disabled due to a conflict.  **Note**  This flag MUST be used only with a forest trust record type of ForestTrustDomainInfo. |
| LSA\_NB\_DISABLED\_ADMIN  0x00000004 | The domain information trust record is disabled by the domain administrator.  **Note**  This flag MUST be used only with a forest trust record type of ForestTrustDomainInfo. |
| LSA\_NB\_DISABLED\_CONFLICT  0x00000008 | The domain information trust record is disabled due to a conflict.  **Note**  This flag MUST be used only with a forest trust record type of ForestTrustDomainInfo. |
| LSA\_FTRECORD\_DISABLED\_REASONS  0x0000FFFF | The domain information trust record is disabled.  **Note**  This set of flags is reserved; for current and future reasons, the trust is disabled. |

**ForestTrustType:**  This value is one of [LSA\_FOREST\_TRUST\_RECORD\_TYPE](#Section_700a91e8a1a44e1b9ad6096b3cf0bef0).

**Time:**  The date and time when this entry was created. It is a 64-bit value that represents the number of 100-nanosecond intervals since January 1, 1601, [**UTC**](#gt_f2369991-a884-4843-a8fa-1505b6d5ece7).

**ForestTrustData:**  An [LSA\_UNICODE\_STRING](#Section_4b35e17e405c4e998ebe8b28f047156f) or [LSA\_FOREST\_TRUST\_DOMAIN\_INFO](#Section_451ac72fe9ad4a4f961fd04a2a5b1515) structure, depending on the value ForestTrustType as specified in the structure definition for LSA\_FOREST\_TRUST\_RECORD.

#### LSA\_FOREST\_TRUST\_RECORD\_TYPE

The LSA\_FOREST\_TRUST\_RECORD\_TYPE enumeration specifies a type of [**forest trust**](#gt_035d9ce5-f117-4251-8d4d-127c462ec4a0) record.

1. typedef enum \_LSA\_FOREST\_TRUST\_RECORD\_TYPE
2. {
3. ForestTrustTopLevelName = 0,
4. ForestTrustTopLevelNameEx = 1,
5. ForestTrustDomainInfo = 2
6. } LSA\_FOREST\_TRUST\_RECORD\_TYPE;

**ForestTrustTopLevelName:** The [**DNS name**](#gt_102a36e2-f66f-49e2-bee3-558736b2ecd5) of the [**trusted forest**](#gt_3b76a71f-9697-4836-9c69-09899b23c21b). The structure used for this record type is equivalent to [LSA\_UNICODE\_STRING (section 2.2.2.3)](#Section_4b35e17e405c4e998ebe8b28f047156f).

**ForestTrustTopLevelNameEx:** The DNS name of the trusted forest. This is the same as **ForestTrustTopLevelName**. The structure used for this record type is equivalent to LSA\_UNICODE\_STRING.

**ForestTrustDomainInfo:** This field specifies a record containing identification and name information.

#### LSA\_FOREST\_TRUST\_BINARY\_DATA

The LSA\_FOREST\_TRUST\_BINARY\_DATA structure is used to communicate a [**forest trust**](#gt_035d9ce5-f117-4251-8d4d-127c462ec4a0) record. This structure is not used in the current version of the protocol.

1. typedef struct \_LSA\_FOREST\_TRUST\_BINARY\_DATA {
2. [range(0,131072)] unsigned long Length;
3. [size\_is(Length)] unsigned char\* Buffer;
4. } LSA\_FOREST\_TRUST\_BINARY\_DATA,
5. \*PLSA\_FOREST\_TRUST\_BINARY\_DATA;

**Length:**  The count of bytes in **Buffer**.[<32>](#Appendix_A_32" \o "Product behavior note 32)

**Buffer:**  The [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) record. If the **Length** field has a value other than 0, this field MUST NOT be NULL.

#### LSA\_FOREST\_TRUST\_DOMAIN\_INFO

The LSA\_FOREST\_TRUST\_DOMAIN\_INFO structure is used to communicate a [**forest trust**](#gt_035d9ce5-f117-4251-8d4d-127c462ec4a0) record corresponding to the LSA\_FOREST\_TRUST\_DOMAIN\_INFO value of ForestTrustDomainInfo.

1. typedef struct \_LSA\_FOREST\_TRUST\_DOMAIN\_INFO {
2. PRPC\_SID Sid;
3. LSA\_UNICODE\_STRING DnsName;
4. LSA\_UNICODE\_STRING NetbiosName;
5. } LSA\_FOREST\_TRUST\_DOMAIN\_INFO,
6. \*PLSA\_FOREST\_TRUST\_DOMAIN\_INFO;

**Sid:**  [**Domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) for the [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4).

**DnsName:**  The [**DNS name**](#gt_102a36e2-f66f-49e2-bee3-558736b2ecd5) of the trusted domain.

**NetbiosName:**  The NetBIOS name of the trusted domain, as specified in [[RFC1088]](https://go.microsoft.com/fwlink/?LinkId=90266).

#### LSA\_FOREST\_TRUST\_INFORMATION

The LSA\_FOREST\_TRUST\_INFORMATION structure is a collection of [LSA\_FOREST\_TRUST\_RECORD (section 2.2.7.21)](#Section_08cf1a65ef7c46d3aa4d558f5135df3d) structures.

1. typedef struct \_LSA\_FOREST\_TRUST\_INFORMATION {
2. [range(0,4000)] unsigned long RecordCount;
3. [size\_is(RecordCount)] PLSA\_FOREST\_TRUST\_RECORD\* Entries;
4. } LSA\_FOREST\_TRUST\_INFORMATION,
5. \*PLSA\_FOREST\_TRUST\_INFORMATION;

**RecordCount:**  A count of elements in the Entries array.[<33>](#Appendix_A_33" \o "Product behavior note 33)

**Entries:**  An array of LSA\_FOREST\_TRUST\_RECORD structures. If the **RecordCount** field has a value other than 0, this field MUST NOT be NULL.

#### LSA\_FOREST\_TRUST\_COLLISION\_RECORD\_TYPE

The LSA\_FOREST\_TRUST\_COLLISION\_RECORD\_TYPE type specifies the type of a collision record in the message.

1. typedef enum \_LSA\_FOREST\_TRUST\_COLLISION\_RECORD\_TYPE
2. {
3. CollisionTdo = 0,
4. CollisionXref,
5. CollisionOther
6. } LSA\_FOREST\_TRUST\_COLLISION\_RECORD\_TYPE;

**CollisionTdo:** A [**forest trust**](#gt_035d9ce5-f117-4251-8d4d-127c462ec4a0) record that a caller attempted to set on a [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4) has suffered a collision with another trusted domain object in [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90), as specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a), section 6.1.6.

**CollisionXref:** A forest trust record that a caller attempted to set on a trusted domain object has suffered a collision with a cross-reference object belonging to the [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) to which the server belongs, as specified in [MS-ADTS], section 6.1.6.

**CollisionOther:** A forest trust record that a caller attempted to set on a trusted domain object has suffered a collision for an unknown reason.

#### LSA\_FOREST\_TRUST\_COLLISION\_RECORD

The LSA\_FOREST\_TRUST\_COLLISION\_RECORD structure is used to communicate [**forest trust**](#gt_035d9ce5-f117-4251-8d4d-127c462ec4a0) collision information. For more information about [**trusted domain objects**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4), see [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6.

1. typedef struct \_LSA\_FOREST\_TRUST\_COLLISION\_RECORD {
2. unsigned long Index;
3. LSA\_FOREST\_TRUST\_COLLISION\_RECORD\_TYPE Type;
4. unsigned long Flags;
5. LSA\_UNICODE\_STRING Name;
6. } LSA\_FOREST\_TRUST\_COLLISION\_RECORD,
7. \*PLSA\_FOREST\_TRUST\_COLLISION\_RECORD;

**Index:**  An ordinal number of a forest trust record in the [**forest trust information**](#gt_8c0b82d9-efec-4723-95a9-8564edf9ba44) supplied by the caller that suffered a collision. For rules about collisions, see sections [3.1.4.7.16](#Section_16be42bce85c4135b183aacb88106306) and [3.1.4.7.16.1](#Section_f0e0eefde53c463bb81686fb21931366).

**Type:**  The type of collision record, as specified in section [2.2.7.26](#Section_afc7d769a31748059f4585d5393b57af).

**Flags:**  A set of bits specifying the nature of the collision. These flags and the rules for generating them are specified in sections 3.1.4.7.16 and 3.1.4.7.16.1.

**Name:**  The name of the existing entity (a top-level name entry, a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) information entry, or a top-level name exclusion entry) that caused the collision.

#### LSA\_FOREST\_TRUST\_COLLISION\_INFORMATION

The LSA\_FOREST\_TRUST\_COLLISION\_INFORMATION structure is used to communicate a set of [LSA\_FOREST\_TRUST\_COLLISION\_RECORD](#Section_32178d2cca744f538264af1906f95011) structures.

1. typedef struct \_LSA\_FOREST\_TRUST\_COLLISION\_INFORMATION {
2. unsigned long RecordCount;
3. [size\_is(RecordCount)] PLSA\_FOREST\_TRUST\_COLLISION\_RECORD\* Entries;
4. } LSA\_FOREST\_TRUST\_COLLISION\_INFORMATION,
5. \*PLSA\_FOREST\_TRUST\_COLLISION\_INFORMATION;

**RecordCount:**  The count of elements in the Entries array.

**Entries:**  An array of LSA\_FOREST\_TRUST\_COLLISION\_RECORD (section 2.2.7.27) structures. If the **RecordCount** field has a value other than zero, this field MUST NOT be NULL.

### Privilege Data Types

#### LSAPR\_POLICY\_PRIVILEGE\_DEF

The LSAPR\_POLICY\_PRIVILEGE\_DEF structure specifies a [**privilege**](#gt_d8092e10-b227-4b44-b015-511bb8178940) definition, which consists of a pairing of a human-readable name with a [**locally unique identifier (LUID)**](#gt_96b64af9-1896-4bde-b988-54d469c5affd).

1. typedef struct \_LSAPR\_POLICY\_PRIVILEGE\_DEF {
2. RPC\_UNICODE\_STRING Name;
3. LUID LocalValue;
4. } LSAPR\_POLICY\_PRIVILEGE\_DEF,
5. \*PLSAPR\_POLICY\_PRIVILEGE\_DEF;

**Name:**  An RPC\_UNICODE\_STRING that contains the privilege name.

**LocalValue:**  This field contains the LUID value assigned locally for efficient representation of the privilege. This value is meaningful only on the system where it was assigned.

#### LSAPR\_PRIVILEGE\_ENUM\_BUFFER

The LSAPR\_PRIVILEGE\_ENUM\_BUFFER structure specifies a collection of [**privilege**](#gt_d8092e10-b227-4b44-b015-511bb8178940) definitions of type [LSAPR\_POLICY\_PRIVILEGE\_DEF](#Section_f36d47375b2f4bc08f29e7b4c71b7401).

1. typedef struct \_LSAPR\_PRIVILEGE\_ENUM\_BUFFER {
2. unsigned long Entries;
3. [size\_is(Entries)] PLSAPR\_POLICY\_PRIVILEGE\_DEF Privileges;
4. } LSAPR\_PRIVILEGE\_ENUM\_BUFFER,
5. \*PLSAPR\_PRIVILEGE\_ENUM\_BUFFER;

**Entries:**  This field contains the number of privileges in the structure.

**Privileges:**  This field contains a set of structures that define the privileges, as specified in section 2.2.8.1. If the **Entries** field has a value other than 0, this field MUST NOT be NULL.

## Directory Service Schema Elements

This protocol is part of the [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) core family of protocols. In order to be fully compliant with Active Directory, an implementation of this protocol must be used in conjunction with the full Active Directory schema, containing all the schema attributes and classes 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), and [[MS-ADSC]](%5bMS-ADSC%5d.pdf#Section_9abb5e97123d4da99557b353ab79b830).

# Protocol Details

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

## Server Details

The Local Security Authority (Domain Policy) Remote Protocol server handles client requests for any of the messages described in section [3.1.4](#Section_2c6f3cf9d7924e8b9af55470f636c20a), and operates on the security policy settings stored on the server. For each message, the behavior of the server while processing messages is described in section 3.1.4.

### Abstract Data Model

The Local Security Authority (Domain Policy) Remote Protocol defines an abstract data model that contains information about three types of objects: [**account objects**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b), [**secret objects**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d), and [**trusted domain objects**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4). In addition, this abstract data model contains the policy object that holds miscellaneous policy settings that are unrelated to any of these three types of objects, but apply to the operation of the host of the server implementation of the protocol. Each object contains a few fields; operations on these fields MUST satisfy the [**ACID**](#gt_44f0cb35-f618-456d-b7b1-25213fed1045) properties [GRAY]. Thus, if fields are defined by structures, it is expected that the entire structure be operated on as a unit.

This data model MUST consist of variables whose values are maintained across system restarts and [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) method invocations and that store those values for retrieval and update, unless otherwise specified.

**Note**  The abstract notation (Public) indicates that this Abstract Data Model element can be directly accessed from outside this protocol, for the purpose of documentary convenience. Such direct access MUST NOT be construed as a relaxation of the security constraints specified within this document; rather, the same authorization decisions that are applied when clients access such data elements using protocol primitives MUST also be applied during direct access of the elements. See section [3.1.1.10](#Section_93798a19a71a4d32a5bcf44e2c49b2ea) for more details.

#### Policy Object Data Model

The policy object contains miscellaneous policy settings. There is one object of this type on the server. This object cannot be deleted, and a new object of this type cannot be created. Its fields, however, can be changed when they adhere to the rules in the specification. The data model is presented here as a collection of structures defined in section [2.2](#Section_4d4678cf32154ecc8dc95a2aaa0e1eb0) to ensure that syntax and other consistency rules are met in the data model.[<34>](#Appendix_A_34" \o "Product behavior note 34)

| Name | Type |
| --- | --- |
| Auditing Log Information | POLICY\_AUDIT\_LOG\_INFO |
| Audit Full Information | POLICY\_AUDIT\_FULL\_QUERY\_INFO |
| Event Auditing Options | LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO |
| Primary Domain Information | LSAPR\_POLICY\_PRIMARY\_DOM\_INFO |
| DNS Domain Information (Public)[<35>](#Appendix_A_35" \o "Product behavior note 35) | LSAPR\_POLICY\_DNS\_DOMAIN\_INFO |
| Account Domain Information | LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO |
| Server Role Information | POLICY\_LSA\_SERVER\_ROLE\_INFO |
| Replica Source Information | LSAPR\_POLICY\_REPLICA\_SRCE\_INFO |
| \* Kerberos Policy Information[<36>](#Appendix_A_36" \o "Product behavior note 36) | POLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO |
| Encrypting File System (EFS) Policy Information[<37>](#Appendix_A_37" \o "Product behavior note 37) | LSAPR\_POLICY\_DOMAIN\_EFS\_INFO |
| Quality of Service Information[<38>](#Appendix_A_38" \o "Product behavior note 38) | POLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO |
| Security Descriptor | LSAPR\_SR\_SECURITY\_DESCRIPTOR |

\* The Kerberos Policy Information abstract data contains the following public ADM elements (whose meaning is described in section [2.2.4.19](#Section_AFCC492012D348E0AB95A8989EBBD41D)):

* **AuthenticationOptions** (Public): Optional flags that affect validations.
* **MaxServiceTicketAge** (Public): The maximum ticket lifetime for a [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) ticket.
* **MaxTicketAge** (Public): The maximum ticket lifetime for a ticket-granting ticket.
* **MaxRenewAge** (Public): The maximum renewable lifetime.
* **MaxClockSkew** (Public): The acceptable clock skew.
* **Reserved**: Reserved for future use.

The server MUST notify the Kerberos protocol [[MS-KILE]](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9) when any field of the Kerberos Policy Information ADM element is changed; see section [3.1.4.4.8](#Section_72900DB3247B479C86F01C5B7727971A) for more details.

The following element also pertains to the Policy Object data model:

* **ComputerNetBIOSName**: This ADM element represents the NetBIOS name of the computer. It is shared with the **ComputerName.NetBIOS** element from [[MS-WKST]](%5bMS-WKST%5d.pdf#Section_5bb08058bc364d3cabebb132228281b7) section 3.2.1.2.

Auditing Log Information is constant information about the state of the auditing system. The server MUST store the following constant information.

* MaximumLogSize = 8192 for non–[**domain controllers (DCs)**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd)
* MaximumLogSize = 20480 for domain controllers
* AuditLogPercentFull = 0
* AuditRetentionPeriod = 8533315
* AuditLogFullShutdownInProgress = FALSE
* TimeToShutdown = 288342
* NextAuditRecordId = 0

Account Domain Information stores information about the [**local account domain**](#gt_5127d055-89b1-49ba-adf0-70470d9b9da0) of the machine. Note that Primary Domain Information is returned to clients who issue [LsarQueryInformationPolicy2](#Section_516f503c0230489db012e650b46b66a2) messages (section 3.1.4.4.3) with PolicyAccountDomainInformation to a domain controller.

For domain-joined machines, Primary Domain Information and DNS Domain Information store information about the domain to which the machine is joined. If the machine is not joined to a domain, these abstract data elements store information about the workgroup the machine is in.

The value of the Server Role Information ADM element is determined by the following series of calls to the local SAM Remote Protocol implementation:

1. Invoke **SamrConnect** ([[MS-SAMR]](%5bMS-SAMR%5d.pdf#Section_4df07fab1bbc452f8e927853a3c7e380) section 3.1.5.1.4), specifying SAM\_SERVER\_CONNECT for the *DesiredAccess* parameter.
2. Invoke **SamrLookupDomainInSamServer** ([MS-SAMR] section 3.1.5.11.1), specifying the **Name** field of the Primary Domain Information ADM element for the *Name* parameter
3. Invoke **SamrOpenDomain** ([MS-SAMR] section 3.1.5.1.5), specifying the *ServerHandle* that was obtained in step 1, DOMAIN\_ALL\_ACCESS for the *DesiredAccess* parameter, and the *DomainId* that was obtained in step 2.
4. Invoke **SamrQueryInformationDomain2** ([MS-SAMR] section 3.1.5.5.1), specifying the *DomainHandle* that was obtained in step 3, and DomainServerRoleInformation for the *DomainInformationClass* parameter.
5. The value obtained in step 4 is then used for the Server Role Information ADM element. If DomainServerRolePrimary is returned, then PolicyServerRolePrimary is used; if DomainServerRoleBackup is returned, PolicyServerRoleBackup is used.
6. Call **SamrCloseHandle** ([MS-SAMR] section 3.1.5.13.1) on the handle from step 3.
7. Call **SamrCloseHandle** on the handle from step 1.

Replica Source Information and Encrypting File System (EFS) Policy Information are obsolete abstract data in this version of the protocol. However, an implementation SHOULD support this data for compatibility with previous versions of this protocol.

Audit Full Information and Quality of Service Information are obsolete abstract data in this version of the protocol. An implementation SHOULD choose not to implement this abstract data model.

A [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) is used during handle open for access check. The content of this security descriptor is implementation-specific, but a server MUST assign a default security descriptor.[<39>](#Appendix_A_39" \o "Product behavior note 39)

If the responder for this protocol is a domain controller, the values of the implementation-specific instantiation of Event Auditing Options and Kerberos Policy Information abstract data MUST converge between the domain controllers in the same [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).[<40>](#Appendix_A_40" \o "Product behavior note 40) There is no requirement on the length of time to reach convergence.

#### Accounts Rights Data Model

Account Rights is composed of two submodels, Privilege and System Access Rights. When used with [**account objects**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b), they can be used separately in messages, as in [LsarEnumeratePrivileges](#Section_e1c6e808de604dedb77d32f71a5a934a) and [LsarGetSystemAccessAccount](#Section_6d656257bfb9419fad578e775bf700cf), or together, as in [LsarAddAccountRights](#Section_354b5a33670547f8adfbe291c633e9a8). The **Name** fields in the following data models are used to identify the [**privilege**](#gt_d8092e10-b227-4b44-b015-511bb8178940) or system access right uniquely.

##### Privilege Data Model

The server MUST maintain a list of [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) that it recognizes. A privilege is defined by a language-independent human-readable name, a [**locally unique identifier (LUID)**](#gt_96b64af9-1896-4bde-b988-54d469c5affd), and a language-dependent description of the privilege. Two different privileges MUST have different names as well as different LUIDs. The list of privileges known by the server SHOULD NOT change unless a major event, such as an operating system upgrade, takes place. The set of names identifying privileges and their LUIDs MUST be the same across all servers running the same revision of the operating system.

| Name | Type |
| --- | --- |
| Name | RPC\_UNICODE\_STRING |
| Locally Unique Identifier | LUID |
| Privilege descriptions in different languages | An array of RPC\_UNICODE\_STRINGs |

The Name and Locally Unique Identifier pair are communicated by the Local Security Authority (Domain Policy) Remote Protocol via the [LSAPR\_PRIVILEGE\_ENUM\_BUFFER](#Section_c0278280b4b64538b3aaeb40f64f42fb) structure.

Privilege Description is communicated by the Local Security Authority (Domain Policy) Remote Protocol via the [LsarLookupPrivilegeDisplayName](#Section_6ae3f7f030ed48c2a9bebe4a4617decd) method.

The data model in this version of the protocol defines the privileges described in the table below. The descriptions that are provided are in English.[<41>](#Appendix_A_41" \o "Product behavior note 41)

| Name | LUID | Privilege description |
| --- | --- | --- |
| SE\_ASSIGNPRIMARYTOKEN\_NAME  "SeAssignPrimaryTokenPrivilege" | {0,3} | Replace a process-level token. |
| SE\_AUDIT\_NAME  "SeAuditPrivilege" | {0,21} | Generate security audits. |
| SE\_BACKUP\_NAME  "SeBackupPrivilege" | {0,17} | Back up files and directories. |
| SE\_CHANGE\_NOTIFY\_NAME  "SeChangeNotifyPrivilege" | {0,23} | Bypass traverse checking. |
| SE\_CREATE\_GLOBAL\_NAME  "SeCreateGlobalPrivilege" | {0,30} | Create global objects. |
| SE\_CREATE\_PAGEFILE\_NAME  "SeCreatePagefilePrivilege" | {0,15} | Create a page file. |
| SE\_CREATE\_PERMANENT\_NAME  "SeCreatePermanentPrivilege" | {0,16} | Create permanent shared objects. |
| SE\_CREATE\_TOKEN\_NAME  "SeCreateTokenPrivilege" | {0,2} | Create a token object. |
| SE\_DEBUG\_NAME  "SeDebugPrivilege" | {0,20} | Debug programs. |
| SE\_ENABLE\_DELEGATION\_NAME  "SeEnableDelegationPrivilege" | {0,27} | Enable computer and user accounts to be [**trusted**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) for delegation. |
| SE\_IMPERSONATE\_NAME  "SeImpersonatePrivilege" | {0,29} | Impersonate a client after authentication. |
| SE\_INC\_BASE\_PRIORITY\_NAME  "SeIncreaseBasePriorityPrivilege" | {0,14} | Increase scheduling priority. |
| SE\_INCREASE\_QUOTA\_NAME  "SeIncreaseQuotaPrivilege" | {0,5} | Adjust memory quotas for a process. |
| SE\_LOAD\_DRIVER\_NAME  "SeLoadDriverPrivilege" | {0,10} | Load and unload device drivers. |
| SE\_LOCK\_MEMORY\_NAME  "SeLockMemoryPrivilege" | {0,4} | Lock pages in memory. |
| SE\_MACHINE\_ACCOUNT\_NAME  "SeMachineAccountPrivilege" | {0,6} | Add workstations to domain. |
| SE\_MANAGE\_VOLUME\_NAME  "SeManageVolumePrivilege" | {0,28} | Manage the files on a volume. |
| SE\_PROF\_SINGLE\_PROCESS\_NAME  "SeProfileSingleProcessPrivilege" | {0,13} | Profile single process. |
| SE\_REMOTE\_SHUTDOWN\_NAME  "SeRemoteShutdownPrivilege" | {0,24} | Force shutdown from a remote system. |
| SE\_RESTORE\_NAME  "SeRestorePrivilege" | {0,18} | Restore files and directories. |
| SE\_SECURITY\_NAME  "SeSecurityPrivilege" | {0,8} | Manage auditing and security log. |
| SE\_SHUTDOWN\_NAME  "SeShutdownPrivilege" | {0,19} | Shut down the system. |
| SE\_SYNC\_AGENT\_NAME  "SeSyncAgentPrivilege" | {0,26} | Synchronize [**directory service**](#gt_c36db657-3138-4d9a-9289-ded5cbb8b40e) data. |
| SE\_SYSTEM\_ENVIRONMENT\_NAME  "SeSystemEnvironment" | {0,22} | Modify firmware environment values. |
| SE\_SYSTEM\_PROFILE\_NAME  "SeSystemProfilePrivilege" | {0,11} | Profile system performance. |
| SE\_SYSTEMTIME\_NAME  "SeSystemtimePrivilege" | {0,12} | Change system time. |
| SE\_TAKE\_OWNERSHIP\_NAME  "SeTakeOwnershipPrivilege" | {0,9} | Take ownership of files or other objects. |
| SE\_TCB\_NAME  "SeTcbPrivilege" | {0,7} | Act as part of the operating system. |
| SE\_UNDOCK\_NAME  "SeUndockPrivilege" | {0,25} | Remove computer from docking station. |
| SE\_CREATE\_SYMBOLIC\_LINK\_NAME  "SeCreateSymbolicLinkPrivilege" | {0,35} | Create symbolic links. |
| SE\_INC\_WORKING\_SET\_NAME  "SeIncreaseWorkingSetPrivilege" | {0,33} | Increase a process working set. |
| SE\_RELABEL\_NAME  "SeRelabelPrivilege" | {0,32} | Modify an object label. |
| SE\_TIME\_ZONE\_NAME  "SeTimeZonePrivilege" | {0,34} | Change time zone. |
| SE\_TRUSTED\_CREDMAN\_ACCESS\_NAME  "SeTrustedCredManAccessPrivilege" | {0,31} | Access Credential Manager as a trusted caller. |

##### System Access Rights Data Model

The server MUST maintain a list of system access rights that it recognizes. A system access right is identified by a bit flag and a name. The name is a human-readable form of a system access right. The flag is a representation of the same system access right for data representation.

Fields:

* Name
* Flag

Two different system accesses MUST have different names and different bit flags.

The list of system access rights that MUST be supported are specified in section [2.2.1.2](#Section_ba5a83c10ffe4a819e915739274c03db).[<42>](#Appendix_A_42" \o "Product behavior note 42)

The following table contains the string name that is associated with each system access right. The string name is used in methods that associate a system access with a particular account and that also specify the system access not by a POLICY\_SYSTEM\_ACCESS\_MODE, but by the string specified in this table.

| Name | Flag |
| --- | --- |
| SeInteractiveLogonRight | POLICY\_MODE\_INTERACTIVE  0x00000001 |
| SeNetworkLogonRight | POLICY\_MODE\_NETWORK  0x00000002 |
| SeBatchLogonRight | POLICY\_MODE\_BATCH  0x00000004 |
| SeServiceLogonRight | POLICY\_MODE\_SERVICE  0x00000010 |
| SeDenyInteractiveLogonRight | POLICY\_MODE\_DENY\_INTERACTIVE  0x00000040 |
| SeDenyNetworkLogonRight | POLICY\_MODE\_DENY\_NETWORK  0x00000080 |
| SeDenyBatchLogonRight | POLICY\_MODE\_DENY\_BATCH  0x00000100 |
| SeDenyServiceLogonRight | POLICY\_MODE\_DENY\_SERVICE  0x00000200 |
| SeRemoteInteractiveLogonRight | POLICY\_MODE\_REMOTE\_INTERACTIVE  0x00000400 |
| SeDenyRemoteInteractiveLogonRight | POLICY\_MODE\_DENY\_REMOTE\_INTERACTIVE  0x00000800 |

#### Account Object Data Model

Inside the Local Security Authority (Domain Policy) Remote Protocol database, the [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b) MUST be represented by four pieces of data as follows.

| Name | Type |
| --- | --- |
| Security Identifier (Public) | RPC\_SID |
| Security Descriptor | LSAPR\_SR\_SECURITY\_DESCRIPTOR |
| Privileges (Public) | LSAPR\_PRIVILEGE\_SET |
| System Access Rights | unsigned int with combination of POLICY\_SYSTEM\_ACCESS\_MODE flags |

The **Security Identifier** field identifies the account object and MUST be present. Two different account objects MUST NOT have the same [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d). The **Security Identifier** field MUST be read-only. Any valid SID can be used to identify an account object.

The **Security Descriptor** field controls access to the account object. Every account object in the Local Security Authority (Domain Policy) Remote Protocol database MUST have a valid [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350). The security descriptor can be queried by calling the [LsarQuerySecurityObject](#Section_6b3291a21265498e8e9df7e28962255e) method and changed by calling the [LsarSetSecurityObject](#Section_d4eb72865f194040a0c1d29136e0e58e) method. The server MUST assign a default security descriptor to every newly created account object, even if the client did not specify a default value.[<43>](#Appendix_A_43" \o "Product behavior note 43)

The **Privileges** field is a potentially empty set of "global" rights granted to the account by the server. Every "right" in the set is a pair of a [**LUIDs**](#gt_96b64af9-1896-4bde-b988-54d469c5affd) and a bitmask of attributes. The right can be controlled by calling the [LsarAddAccountRights](#Section_354b5a33670547f8adfbe291c633e9a8), [LsarAddPrivilegesToAccount](#Section_8a542f26243d4341ada58fed194bfcf8), [LsarRemoveAccountRights](#Section_33613a1765c2403d83e9edbff71bc3de), and [LsarRemovePrivilegesFromAccount](#Section_e92d5d073ded4d5299c63057312a37b3) methods. Because there are no "negative" rights, the order of rights in the set is not relevant and the server MUST NOT associate any special semantics with the order of rights.

The **System Access Rights** field is a bitmask of flags indicating the system access of the account.

This field can be set to 0.

If the responder for this protocol is a [**domain controller**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), the values of the implementation-specific instantiation of this abstract data model MUST converge between the domain controller in the same [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).[<44>](#Appendix_A_44" \o "Product behavior note 44) There is no requirement on the length of time to reach convergence.

#### Secret Object Data Model

Inside the Local Security Authority (Domain Policy) Remote Protocol database, a [**secret object**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d) is represented by the following pieces of data.

| Name | Type | Attribute name |
| --- | --- | --- |
| Name | RPC\_UNICODE\_STRING | ldapDisplayName ([[MS-ADA1]](%5bMS-ADA1%5d.pdf#Section_19528560f41e4623a406dabcfff0660f) section 2.356) |
| Security Descriptor | LSAPR\_SR\_SECURITY\_DESCRIPTOR | securityIdentifier ([[MS-ADA3]](%5bMS-ADA3%5d.pdf#Section_4517e8353ee644d4bb95a94b6966bfb0) section 2.237) |
| Old Set Time | LARGE\_INTEGER | priorSetTime ([MS-ADA3] section 2.159) |
| Old Value | binary data | priorValue ([MS-ADA3] section 2.160) |
| New Set Time | LARGE\_INTEGER | lastSetTime ([MS-ADA1] section 2.353) |
| New Value | binary data | currentValue ([MS-ADA1] section 2.139) |

The **Name** field uniquely identifies the secret by using a Unicode string. Two different secrets MUST have different names (the comparison is case-sensitive). The **Name** field MUST be read-only. To be considered valid, the length of the name in bytes MUST be even; it MUST be greater than 0 and less than 0x101. The secret name MUST NOT contain the "\" character. Special values of the **Name** field indicate secret types. The different secret types are as follows:

* Global
* Local
* Trusted Domain
* System

The following rules govern secret type assignments.

The term "starts with" literally means "must have a nonzero number of characters following the prefix". Names consisting of only a reserved prefix are invalid.

The following table indicates the secret name pattern and the associated secret type.

| Secret name or name pattern | Type of secret |
| --- | --- |
| Starts with "G$$" | Trusted domain |
| Starts with "G$" | Global |
| Starts with "L$" | Local |
| Starts with "M$" | System |
| Starts with "\_sc\_" | System |
| Starts with "NL$" | System |
| Starts with "RasDialParams" | Local |
| Starts with "RasCredentials" | Local |
| Equal to "$MACHINE.ACC" | System |
| Equal to "SAC" | Local |
| Equal to "SAI" | Local |
| Equal to "SANSC" | Local |

The type of a secret defines the access and availability boundary for a given secret object.

System Secret: Cannot be accessed by any clients.

Local Secret: Can be accessed only by a client that is on the same machine as the server.

Global Secret: Replicates between [**domain controllers**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) in the same [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), allowing each domain controller to be able to respond to secret requests of this type.

Trusted Domain Secret: Used with [**trusted domain objects**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4) to store [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) passwords. Trusted domain secrets also replicate between domain controllers in the same domain.[<45>](#Appendix_A_45" \o "Product behavior note 45)

The [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) field controls access to the secret object. Every secret object in the Local Security Authority (Domain Policy) Remote Protocol database that has Local Secret type MUST have a valid security descriptor. The security descriptor of Local Secret objects can be queried by calling the [LsarQuerySecurityObject (section 3.1.4.9.1)](#Section_6b3291a21265498e8e9df7e28962255e) method and changed by calling the [LsarSetSecurityObject (section 3.1.4.9.2)](#Section_d4eb72865f194040a0c1d29136e0e58e) method. The server MUST assign a default security descriptor to every newly created secret object, even if the client did not specify a default value.[<46>](#Appendix_A_46" \o "Product behavior note 46)

The value of a secret is a byte BLOB. Depending on the caller's choices, the server stores 0, 1, or 2 values for the secret, the 2 values being "current" and "previous" and 1 value being either "current" or "previous". Both versions of the secret's value are accompanied by a 64-bit time stamp in [**Coordinated Universal Time (UTC)**](#gt_f2369991-a884-4843-a8fa-1505b6d5ece7), sometimes referred to as Greenwich Mean Time, in units of 100 nanoseconds since January 1, 1601.

#### Trusted Domain Object Data Model

An implementer must read [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6 to understand the role of [**trusts**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) in [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) and to understand the data model in this specification.

Inside the Local Security Authority (Domain Policy) Remote Protocol database, a [**trusted domain object (TDO)**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4) is represented by the following table. Each abstract data field listed in the Name column of the table contains a link to the appropriate section in [MS-ADTS] section 6.1.6.7. See these sections for detailed information, including how each abstract data field is mapped to an Active Directory attribute. The Type column lists the data type for its corresponding abstract data field. The Attribute Name column lists the ldapDisplayName and a link to the appropriate section in [[MS-ADA1]](%5bMS-ADA1%5d.pdf#Section_19528560f41e4623a406dabcfff0660f), [[MS-ADA2]](%5bMS-ADA2%5d.pdf#Section_e20ebc4e528540bab3bdffcb81c2783e), or [[MS-ADA3]](%5bMS-ADA3%5d.pdf#Section_4517e8353ee644d4bb95a94b6966bfb0) for the corresponding abstract data field stored in Active Directory.

| Name | Type | Attribute name |
| --- | --- | --- |
| Name ([MS-ADTS] section 6.1.6.7.13) | RPC\_UNICODE\_STRING | trustPartner ([MS-ADA3] section 2.325) |
| Flat Name ([MS-ADTS] section 6.1.6.7.1) | RPC\_UNICODE\_STRING | flatName ([MS-ADA1] section 2.232) |
| Security Identifier ([MS-ADTS] section 6.1.6.7.8) | RPC\_SID | securityIdentifier ([MS-ADA3] section 2.237) |
| Trust Type ([MS-ADTS] section 6.1.6.7.15) | unsigned int (as specified in section [2.2.7.9](#Section_f28f42b7173c4cda98093fe4a5213ab3) TrustType) | trustType ([MS-ADA3] section 2.327) |
| Trust Direction ([MS-ADTS] section 6.1.6.7.12) | unsigned int (as specified in section 2.2.7.9 TrustDirection) | trustDirection ([MS-ADA3] section 2.323) |
| Trust Attributes ([MS-ADTS] section 6.1.6.7.9) | unsigned int (as specified in section 2.2.7.9 TrustAttributes) | trustAttributes ([MS-ADA3] section 2.320) |
| Posix Offset ([MS-ADTS] section 6.1.6.7.14) | [TRUSTED\_POSIX\_OFFSET\_INFO](#Section_b091ee7ef5c34b4885671b08ea002221) | trustPosixOffset ([MS-ADA3] section 2.326) |
| Trust Incoming Passwords ([MS-ADTS] section 6.1.6.7.10) | Array of [LSAPR\_AUTH\_INFORMATION](#Section_cedb0d1bc7c0448099fc279b06f22a0c) | trustAuthIncoming ([MS-ADA3] section 2.321) |
| Trust Outgoing Passwords ([MS-ADTS] section 6.1.6.7.11) | Array of LSAPR\_AUTH\_INFORMATION | trustAuthOutgoing ([MS-ADA3] section 2.322) |
| Supported Encryption Types ([MS-ADTS] section 6.1.6.7.3) | [TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES](#Section_7c519a643dc14be6a17d76817cff6e39) | msDS-SupportedEncryptionTypes ([MS-ADA2] section 2.464) |
| Forest Trust Information ([MS-ADTS] section 6.1.6.7.4) | [LSA\_FOREST\_TRUST\_INFORMATION](#Section_2993ffabc0c846439a794ff7d31922dc) | msDS-TrustForestTrustInfo ([MS-ADA2] section 2.479) |
| Security Descriptor ([MS-ADTS] section 6.1.6.7.5) | [LSAPR\_SR\_SECURITY\_DESCRIPTOR](#Section_5564065e3f3d4481a385367cc9b042c4) | nTSecurityDescriptor ([MS-ADA3] section 2.37) |

The following citation contains a timeline of when each information value was introduced.[<47>](#Appendix_A_47" \o "Product behavior note 47)

#### Configuration Settings

##### Block Anonymous Access to Objects

| Name | Type |
| --- | --- |
| LsaRestrictAnonymous | Boolean |

The LsaRestrictAnonymous setting is used to restrict the ability of anonymous requestors to query or modify security-sensitive data.[<48>](#Appendix_A_48" \o "Product behavior note 48) See sections [3.1.4.4.1](#Section_9456a9637c214710af77d0a2f5a72d6b), [3.1.4.5.1](#Section_841E32115BE44B509F112D4941C40A30), [3.1.4.5.2](#Section_86f5e73b98c4423489cbd9ff5f327b73), [3.1.4.5.3](#Section_355e2952abe447c396d9a2f4bd01cf3d), [3.1.4.5.10](#Section_920f27bbde33418bb6a9add38bcb4f57), [3.1.4.5.12](#Section_33613a1765c2403d83e9edbff71bc3de), [3.1.4.6.1](#Section_35A984A1D0024D60946DB557FF4C46E0), [3.1.4.6.2](#Section_8bf25269014f43fdb80f7a59a4883451), and [3.1.4.6.6](#Section_b46f3725d3de46b78245a14edeb278a1) for information on how message processing is affected with this setting. The server message-processing behavior MUST always reflect the current value of this setting.

This setting MUST be persisted across protocol and system restarts.

#### LsaContextHandle Data Model

This protocol is based largely on the use of [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handles to maintain session state between the client and the server. The basic context-handle programming model is described in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 6.1.6. Also see sections 3.2.3.1.9 and 3.3.1.4.1 in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

The server MUST maintain the following data elements for each context handle that is returned to a client.

| Name | Type |
| --- | --- |
| GrantedAccess | ACCESS\_MASK |
| HandleType | HandleType MUST be one of the following:   * Policy * Account * Secret * Trusted Domain |
| Object | A reference to an object in the database that has the type specified in HandleType. |

#### Attribute Listing

The following attributes are referenced by this protocol (listed by ldapDisplayName). For a normative description of the syntax, see [[MS-ADA1]](%5bMS-ADA1%5d.pdf#Section_19528560f41e4623a406dabcfff0660f), [[MS-ADA2]](%5bMS-ADA2%5d.pdf#Section_e20ebc4e528540bab3bdffcb81c2783e), and [[MS-ADA3]](%5bMS-ADA3%5d.pdf#Section_4517e8353ee644d4bb95a94b6966bfb0).

* currentValue
* flatName
* lastSetTime
* ldapDisplayName
* msDS-AllUsersTrustQuota
* msDS-PerUserTrustQuota
* msDS-PerUserTrustTombstonesQuota
* msDS-SupportedEncryptionTypes
* msDS-TrustForestTrustInfo
* priorSetTime
* priorValue
* securityIdentifier
* trustAuthIncoming
* trustAuthOutgoing
* trustDirection
* trustPartner
* trustPosixOffset
* trustType
* unicodePwd

#### Object Class Listing

The following classes are referenced by this protocol (listed by ldapDisplayName). For a normative description of these classes, see [[MS-ADSC]](%5bMS-ADSC%5d.pdf#Section_9abb5e97123d4da99557b353ab79b830).

* secret
* trustedDomain

#### Access for Public Abstract Data Model Elements

As described in section [3.1.1](#Section_0877fdc4184f40b0b378f50d6647d23e), direct access (query or set) of data elements tagged as "(Public)" MUST use the same authorization policies, enforced as if the elements were being accessed via the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331)-based protocol methods in this document. The calling patterns described in section [1.3](#Section_d0d689888b6e4ea8a05029c3f28234e1) provide an overview for understanding the basic flow of the query and set patterns. Section [3.1.1.10.1](#Section_66d6683b853c4af5b6c3dea38f0043a4) provides detailed examples for the Policy Object Data Model (section [3.1.1.1](#Section_c939c70e8cf04d90a9cc1c5002951723)); the other object types use similar patterns.

The following table describes the level of access that MUST be enforced during direct access of the described public ADM elements.

| Object type | DesiredAccess required for Query pattern | DesiredAccess required for Set pattern |
| --- | --- | --- |
| Policy (section 3.1.1.1) | POLICY\_VIEW\_AUDIT\_INFORMATION | POLICY\_GET\_PRIVATE\_INFORMATION | POLICY\_VIEW\_LOCAL\_INFORMATION | READ\_CONTROL | POLICY\_TRUST\_ADMIN | POLICY\_CREATE\_ACCOUNT | POLICY\_CREATE\_SECRET | POLICY\_CREATE\_PRIVILEGE | POLICY\_SET\_DEFAULT\_QUOTA\_LIMITS | POLICY\_SET\_AUDIT\_REQUIREMENTS | POLICY\_AUDIT\_LOG\_ADMIN | POLICY\_SERVER\_ADMIN | READ\_CONTROL |
| Account (section [3.1.1.3](#Section_d1a82c1b4d2b456fb4dfc70b461f3d42)) | ACCOUNT\_VIEW | READ\_CONTROL | ACCOUNT\_ADJUST\_PRIVILEGES | ACCOUNT\_ADJUST\_QUOTAS | ACCOUNT\_ADJUST\_SYSTEM\_ACCESS | READ\_CONTROL |
| Secret (section [3.1.1.4](#Section_483f1b6e7b1443419ab29b99c01f896e)) | SECRET\_QUERY\_VALUE | READ\_CONTROL | SECRET\_SET\_VALUE | READ\_CONTROL |
| TrustedDomain (section [3.1.1.5](#Section_0228f75e9725479cb4cd1cacd667343c)) | TRUSTED\_QUERY\_DOMAIN\_NAME | READ\_CONTROL | TRUSTED\_SET\_CONTROLLERS | TRUSTED\_SET\_POSIX | READ\_CONTROL |

##### Example Patterns for Direct Access of Policy Object ADM Elements

###### Query Pattern for Policy Object ADM

Direct querying of any of the (Public) ADM elements listed in section [3.1.1](#Section_0877FDC4184F40B0B378F50D6647D23E) MUST be performed as follows:

1. The client MUST invoke [LsarOpenPolicy2 (section 3.1.4.4.1)](#Section_9456a9637c214710af77d0a2f5a72d6b), specifying NULL for the *SystemName* parameter and POLICY\_VIEW\_AUDIT\_INFORMATION | POLICY\_VIEW\_LOCAL\_INFORMATION | POLICY\_GET\_PRIVATE\_INFORMATION | READ\_CONTROL for the *DesiredAccess* parameter.
2. The client MUST invoke [LsarQueryInformationPolicy2 (section 3.1.4.4.3)](#Section_516f503c0230489db012e650b46b66a2), specifying the policy handle obtained in step 1 for the *PolicyHandle* parameter and PolicyDnsDomainInformation for the *InformationClass* parameter.
3. The client MUST invoke [LsarClose (section 3.1.4.9.4)](#Section_99dd2d7ab0fc4c6d837a2b4d342383ae), specifying the policy handle obtained in step 1 for the *ObjectHandle* parameter.
4. The ADM elements of interest are then read from the **LSAPR\_POLICY\_INFORMATION.PolicyDnsDomainInfo** structure (section [2.2.4.14](#Section_3e15a02e25d346aa9c608def03c824d2)) that was returned in step 2.

###### Set Pattern for Policy Object ADM

Direct setting of any of the (Public) ADM elements listed in section [3.1.1](#Section_0877FDC4184F40B0B378F50D6647D23E) MUST be performed as follows:

1. The client MUST invoke [LsarOpenPolicy2 (section 3.1.4.4.1)](#Section_9456a9637c214710af77d0a2f5a72d6b), specifying NULL for the *SystemName* parameter and POLICY\_TRUST\_ADMIN | POLICY\_CREATE\_ACCOUNT | POLICY\_CREATE\_SECRET | POLICY\_CREATE\_PRIVILEGE | POLICY\_SET\_DEFAULT\_QUOTA\_LIMITS | POLICY\_SET\_AUDIT\_REQUIREMENTS | POLICY\_AUDIT\_LOG\_ADMIN | POLICY\_SERVER\_ADMIN | READ\_CONTROL for the *DesiredAccess* parameter.
2. The client MUST invoke [LsarQueryInformationPolicy2 (section 3.1.4.4.3)](#Section_516f503c0230489db012e650b46b66a2), specifying the policy handle obtained in step 1 for the *PolicyHandle* parameter and PolicyDnsDomainInformation for the *InformationClass* parameter.
3. The client MUST set the ADM elements of interest in the **LSAPR\_POLICY\_INFORMATION.PolicyDnsDomainInfo** structure (section [2.2.4.14](#Section_3e15a02e25d346aa9c608def03c824d2)) that was returned in step 2 to the desired new values, leaving the remaining elements unmodified.
4. The client MUST invoke [LsarSetInformationPolicy2 (section 3.1.4.4.5)](#Section_fc14e9aea26e4031809ea908dd3e13a3), specifying the policy handle obtained in step 1 for the *PolicyHandle* parameter.
5. The client MUST invoke [LsarClose (section 3.1.4.9.4)](#Section_99dd2d7ab0fc4c6d837a2b4d342383ae), specifying the policy handle obtained in step 1 for the *ObjectHandle* parameter.

### Timers

No protocol timers are required other than those internal ones used in [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) to implement resiliency to network outages, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

### Initialization

The server MUST start listening on the well-known named pipe for the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface, as specified in section [2.1](#Section_64ea7ac432ef44f6ab51ea2b5a1c2390).

The **ComputerNetBIOSName** element (specified in section [3.1.1.1](#Section_C939C70E8CF04D90A9CC1C5002951723)) MUST be copied into the **DomainName** field in Account Domain Information (also specified in section 3.1.1.1).

### Message Processing Events and Sequencing Rules

This section contains detailed information about each protocol message and the steps taken by the server to process caller requests.[<49>](#Appendix_A_49" \o "Product behavior note 49)[<50>](#Appendix_A_50" \o "Product behavior note 50)[<51>](#Appendix_A_51" \o "Product behavior note 51)

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [LsarClose](#Section_99dd2d7ab0fc4c6d837a2b4d342383ae) | This method closes an open handle.  Opnum: 0 |
| Opnum1NotUsedOnWire | Opnum: 1 |
| [LsarEnumeratePrivileges](#Section_e1c6e808de604dedb77d32f71a5a934a) | This method is invoked to enumerate all [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) known to the system.  Opnum: 2 |
| [LsarQuerySecurityObject](#Section_6b3291a21265498e8e9df7e28962255e) | This method is invoked to query security information that is assigned to a database object. It returns the [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) of the object.  Opnum: 3 |
| [LsarSetSecurityObject](#Section_d4eb72865f194040a0c1d29136e0e58e) | This method is invoked to set a security descriptor on an object.  Opnum: 4 |
| Opnum5NotUsedOnWire | Opnum: 5 |
| [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) | This method is exactly the same as [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b), except that the *SystemName* parameter in this function, because of its syntactic definition, contains only one character instead of a full string.  Opnum: 6 |
| [LsarQueryInformationPolicy](#Section_3564ba7084ea4f04a9dcdede9f96a8bf) | This method is invoked to query values representing the server's information policy.  Opnum: 7 |
| [LsarSetInformationPolicy](#Section_8a82ce8168e142da88a751096dcde022) | This method is invoked to set some policy on the server.  Opnum: 8 |
| Opnum9NotUsedOnWire | Opnum: 9 |
| [LsarCreateAccount](#Section_841e32115be44b509f112d4941c40a30) | This method is invoked to create a new [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b) in the server's database.  Opnum: 10 |
| [LsarEnumerateAccounts](#Section_86f5e73b98c4423489cbd9ff5f327b73) | This method is invoked to request a list of account objects in the server's database.  Opnum: 11 |
| [LsarCreateTrustedDomain](#Section_373a4b1e1e8d45729c250bd7b045d3a3) | This method is invoked to create an object of type [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4) in the server's database.  Opnum: 12 |
| [LsarEnumerateTrustedDomains](#Section_3de62a51861e4373ae2ff3433cc10106) | This method is invoked to request a list of [**TDOs**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4) in the server's database.  Opnum: 13 |
| Lsar\_LSA\_TM\_14 | Opnum: 14 |
| Lsar\_LSA\_TM\_15 | Opnum: 15 |
| [LsarCreateSecret](#Section_35a984a1d0024d60946db557ff4c46e0) | This method is invoked to create a new [**secret object**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d) in the server's database.  Opnum: 16 |
| [LsarOpenAccount](#Section_355e2952abe447c396d9a2f4bd01cf3d) | This method is invoked to obtain a handle to an account object.  Opnum: 17 |
| [LsarEnumeratePrivilegesAccount](#Section_0e99240e58574b3d85c4d24f3155f6d4) | This method is invoked to retrieve a list of privileges granted to an account on the server.  Opnum: 18 |
| [LsarAddPrivilegesToAccount](#Section_8a542f26243d4341ada58fed194bfcf8) | This method is invoked to add new privileges to an existing account object.  Opnum: 19 |
| [LsarRemovePrivilegesFromAccount](#Section_e92d5d073ded4d5299c63057312a37b3) | This method is invoked to remove privileges from an account object.  Opnum: 20 |
| Opnum21NotUsedOnWire | Opnum: 21 |
| Opnum22NotUsedOnWire | Opnum: 22 |
| [LsarGetSystemAccessAccount](#Section_6d656257bfb9419fad578e775bf700cf) | This method is invoked to retrieve system access account flags for an account object.  Opnum: 23 |
| [LsarSetSystemAccessAccount](#Section_3d73874ca2d7456aac220e7ece767a28) | This method is invoked to set system access account flags for an account object.  Opnum: 24 |
| [LsarOpenTrustedDomain](#Section_c59462aa9dca49e09ff1a5a009f0c64c) | This method is invoked to obtain a handle to a TDO.  Opnum: 25 |
| [LsarQueryInfoTrustedDomain](#Section_e74460c7db0345c3ac3ca72a840e4943) | This method is invoked to retrieve information on a TDO.  Opnum: 26 |
| [LsarSetInformationTrustedDomain](#Section_9ea46cefcc724109ba1391eda6b713bc) | This method is invoked to set information on a TDO.  Opnum: 27 |
| [LsarOpenSecret](#Section_8bf25269014f43fdb80f7a59a4883451) | This method is invoked to obtain a handle to an existing secret object.  Opnum: 28 |
| [LsarSetSecret](#Section_21c1a153032c4869afc9186b2346dfab) | This method is invoked to set the current and old values of the secret object.  Opnum: 29 |
| [LsarQuerySecret](#Section_e36cfffafd53437ea5a71a95cfdda4c1) | This method is invoked to retrieve the current and old (or previous) value of the secret object.  Opnum: 30 |
| [LsarLookupPrivilegeValue](#Section_1f36a524be2640399047ad71899d0d8c) | This method is invoked to map the name of a privilege into the [**LUID**](#gt_96b64af9-1896-4bde-b988-54d469c5affd) by which the privilege is known on the server.  Opnum: 31 |
| [LsarLookupPrivilegeName](#Section_b4ae2755389741178a832b98874e3982) | This method is invoked to map the LUID of a privilege into the string name by which the privilege is known on the server.  Opnum: 32 |
| [LsarLookupPrivilegeDisplayName](#Section_6ae3f7f030ed48c2a9bebe4a4617decd) | This method is invoked to map the name of a privilege into a display text string in the caller's language.  Opnum: 33 |
| [LsarDeleteObject](#Section_8d0aa2dc22b64bc3b5d279b4b0ad7bce) | This method is invoked to delete an open account, secret, or TDO.  Opnum: 34 |
| [LsarEnumerateAccountsWithUserRight](#Section_9c6ec6db534b41d5bbacfaa9ad31b380) | This method is invoked to return a list of account objects that have the user right equal to the passed-in value.  Opnum: 35 |
| [LsarEnumerateAccountRights](#Section_920f27bbde33418bb6a9add38bcb4f57) | This method is invoked to retrieve a list of rights that are associated with an existing account.  Opnum: 36 |
| [LsarAddAccountRights](#Section_354b5a33670547f8adfbe291c633e9a8) | This method is invoked to add new rights to an account object.  Opnum: 37 |
| [LsarRemoveAccountRights](#Section_33613a1765c2403d83e9edbff71bc3de) | This method is invoked to remove rights from an account object.  Opnum: 38 |
| [LsarQueryTrustedDomainInfo](#Section_862cdc4f79e349738fd32e7836010eb5) | This method is invoked to retrieve information on a TDO.  Opnum: 39 |
| [LsarSetTrustedDomainInfo](#Section_d541b40a06ad4b3bbbfa5cf51a1f02d9) | This method is invoked to set information on a TDO.  Opnum: 40 |
| [LsarDeleteTrustedDomain](#Section_633788c91e984555bc41d78a0de0f4a4) | This method is invoked to delete a TDO.  Opnum: 41 |
| [LsarStorePrivateData](#Section_b79c94fed7174ecf963ca200682921dc) | This method is invoked to store a secret value.  Opnum: 42 |
| [LsarRetrievePrivateData](#Section_b46f3725d3de46b78245a14edeb278a1) | This method is invoked to retrieve a secret value.  Opnum: 43 |
| LsarOpenPolicy2 | This method opens a context handle to the [**RPC server**](#gt_ae65dac0-cd24-4e83-a946-6d1097b71553).  Opnum: 44 |
| Lsar\_LSA\_TM\_45 | Opnum: 45 |
| [LsarQueryInformationPolicy2](#Section_516f503c0230489db012e650b46b66a2) | This method is identical to LsarQueryInformationPolicy.  Opnum: 46 |
| [LsarSetInformationPolicy2](#Section_fc14e9aea26e4031809ea908dd3e13a3) | This method is identical to LsarSetInformationPolicy.  Opnum: 47 |
| [LsarQueryTrustedDomainInfoByName](#Section_5778eba18f9b4696b30434c58edbc5b1) | This method is invoked to retrieve information on a TDO by its string name.  Opnum: 48 |
| [LsarSetTrustedDomainInfoByName](#Section_7a243aac9d5942eaba2754775c4f1573) | This method is invoked to set information on a TDO by its string name.  Opnum: 49 |
| [LsarEnumerateTrustedDomainsEx](#Section_14e37cf7b090497ca2e297e8425532a2) | This method is invoked to enumerate TDOs in the server's database.  Opnum: 50 |
| [LsarCreateTrustedDomainEx](#Section_6817095dd3414d0393d7e9bdca2d3eef) | This method is invoked to create a new TDO.  Opnum: 51 |
| Opnum52NotUsedOnWire | Opnum: 52 |
| [LsarQueryDomainInformationPolicy](#Section_833ec261c4f649a4be211dd31a9525c0) | This method is invoked to retrieve policy settings pertaining to the current [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).  Opnum: 53 |
| [LsarSetDomainInformationPolicy](#Section_72900db3247b479c86f01c5b7727971a) | This method is invoked to change policy settings pertaining to the current domain.  Opnum: 54 |
| [LsarOpenTrustedDomainByName](#Section_62ba989fbd1a459cb402bf315d44c6e1) | This method is invoked to open a TDO handle by supplying the name of the trusted domain.  Opnum: 55 |
| Opnum56NotUsedOnWire | Opnum: 56 |
| Lsar\_LSA\_TM\_57 | Opnum: 57 |
| Lsar\_LSA\_TM\_58 | Opnum: 58 |
| [LsarCreateTrustedDomainEx2](#Section_cc86a55db61948fd998a65cca15efeb9) | This method is invoked to create a new TDO.  Opnum: 59 |
| Opnum60NotUsedOnWire | Opnum: 60 |
| Opnum61NotUsedOnWire | Opnum: 61 |
| Opnum62NotUsedOnWire | Opnum: 62 |
| Opnum63NotUsedOnWire | Opnum: 63 |
| Opnum64NotUsedOnWire | Opnum: 64 |
| Opnum65NotUsedOnWire | Opnum: 65 |
| Opnum66NotUsedOnWire | Opnum: 66 |
| Opnum67NotUsedOnWire | Opnum: 67 |
| Lsar\_LSA\_TM\_68 | Opnum: 68 |
| Opnum69NotUsedOnWire | Opnum: 69 |
| Opnum70NotUsedOnWire | Opnum: 70 |
| Opnum71NotUsedOnWire | Opnum: 71 |
| Opnum72NotUsedOnWire | Opnum: 72 |
| [LsarQueryForestTrustInformation](#Section_fbc7a303997047a182cf0bd635a35bd1) | This method is invoked to retrieve information on a [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) relationship with another [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62).  Opnum: 73 |
| [LsarSetForestTrustInformation](#Section_16be42bce85c4135b183aacb88106306) | This method is invoked to establish a trust relationship with another forest by attaching a set of records called the [**forest trust information**](#gt_8c0b82d9-efec-4723-95a9-8564edf9ba44) to the TDO.  Opnum: 74 |

The following citation contains a timeline of when each method value was introduced.[<52>](#Appendix_A_52" \o "Product behavior note 52)

**Note**  Gaps in the [**opnum**](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7) numbering sequence represent opnums of methods that are specified in [[MS-LSAT]](%5bMS-LSAT%5d.pdf#Section_1ba21e6fd8a9462c91534375f2020894), or opnums that MUST NOT be used over the wire.[<53>](#Appendix_A_53" \o "Product behavior note 53)

**Note**  Exceptions MUST NOT be thrown beyond those thrown by the underlying [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)), unless otherwise specified.

The return values of all methods MUST conform to the specification of NTSTATUS, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90) section 2.3. Specific return values for normative processing conditions are specified in this document in the subsections of this section.

Unless otherwise specified, all negative values returned by an implementation are treated equivalently by the client as a message processing error. Unless otherwise specified, all non-negative values returned by an implementation are treated equivalently by the client as a success (of message processing).

Return values for implementation-specific conditions are left to the implementer's discretion, subject to the constraints specified in [MS-ERREF]. For example, an implementation can re-use an existing value in [MS-ERREF], such as 0xC0000017 (no memory).

All methods in this protocol MUST perform data validation (as specified in section [3.1.4.10](#Section_ba5f1e31b2a742bd9a903650a6e5f6f5)) for all parameters that are specified as input parameters. If data validation fails for some reason, processing MUST end, and the server MUST respond back with a failure.

In the following sections, the first general idea behind the common operations is explained in sections [3.1.4.1](#Section_879d309fe4d4424b98482913d14994aa), [3.1.4.2](#Section_e6003a157a4a46e6a763fe40522243ce), and [3.1.4.3](#Section_bfa65c4ab83741cf819048671f803d4f). The methods are grouped by functionality: policies, accounts, secrets, trusted domains, privileges, and common object methods. Section 3.1.4.10 explains the data validation rules.

#### Obtaining Handles

The Local Security Authority (Domain Policy) Remote Protocol recognizes four types of handles: Policy, Account, Secret, and Trusted Domain. A handle of each type can be obtained only by calling one of a well-defined set of methods. These handles are listed in the following table.

| Handle type | Methods that return this type of handle |
| --- | --- |
| Policy | [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a)  [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b) |
| Account | [LsarCreateAccount](#Section_841e32115be44b509f112d4941c40a30)  [LsarOpenAccount](#Section_355e2952abe447c396d9a2f4bd01cf3d) |
| Secret | [LsarCreateSecret](#Section_35a984a1d0024d60946db557ff4c46e0)  [LsarOpenSecret](#Section_8bf25269014f43fdb80f7a59a4883451) |
| Trusted Domain | [LsarCreateTrustedDomain](#Section_373a4b1e1e8d45729c250bd7b045d3a3)  [LsarOpenTrustedDomain](#Section_c59462aa9dca49e09ff1a5a009f0c64c)  [LsarCreateTrustedDomainEx](#Section_6817095dd3414d0393d7e9bdca2d3eef)  [LsarOpenTrustedDomainByName](#Section_62ba989fbd1a459cb402bf315d44c6e1)  [LsarCreateTrustedDomainEx2](#Section_cc86a55db61948fd998a65cca15efeb9) |

The server MUST keep track of all handles of each type that every caller opens, from the moment of creation until the handle has been closed (by calling [LsarClose](#Section_99dd2d7ab0fc4c6d837a2b4d342383ae) or [LsarDeleteObject](#Section_8d0aa2dc22b64bc3b5d279b4b0ad7bce)) or until the client disconnects.

Upon receipt of a handle parameter, the server MUST check to see that the handle is one of the valid handles of a type relevant for that operation; if the handle is not valid, the server MUST fail the request by returning STATUS\_INVALID\_HANDLE.

The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol provides a mechanism to clean up any resources related to a context handle if a client that is holding the context handle exits, dies, disconnects, or reboots. See section [3.1.6.1](#Section_2184a151ce744e1e8ba725384f51dbf8) for this protocol's context handle rundown specification.

#### Access Rights and Access Checks

Methods in this protocol perform one or more of the access checks that are specified in the following sections.

| Access check (section) | Methods that use it |
| --- | --- |
| [3.1.4.2.1](#Section_e5e1e32e4066435db669044fe997eaf7) | **LsarOpenPolicy**  **LsarOpenPolicy2**  **LsarCreateAccount**  **LsarOpenAccount**  **LsarCreateSecret**  **LsarOpenSecret**  **LsarCreateTrustedDomain**  **LsarOpenTrustedDomain**  **LsarCreateTrustedDomainEx**  **LsarOpenTrustedDomainByName**  **LsarCreateTrustedDomainEx2** |
| [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) | **LsarQueryInformationPolicy2**  **LsarQueryInformationPolicy**  **LsarSetInformationPolicy2**  **LsarSetInformationPolicy**  **LsarQueryDomainInformationPolicy**  **LsarSetDomainInformationPolicy**  **LsarCreateAccount**  **LsarEnumerateAccounts**  **LsarEnmeratePrivilegesAccount**  **LsarAddPrivilegesToAccount**  **LsarRemovePrivilegesFromAccount**  **LsarGetSystemAccessAccount**  **LsarSetSystemAccessAcount**  **LsarEnumerateAccountsWithUserRight**  **LsarEnumerateAccountRights**  **LsarAddAccountRights**  **LsarRemoveAccountRights**  **LsarCreateSecret**  **LsarSetSecret**  **LsarQuerySecret**  **LsarStorePrivateData**  **LsarRetrievePrivateData**  **LsarQueryTrustedDomainInfo**  **LsarDeleteTrustedDomain**  **LsarQueryTrustedDomainInfoByName**  **LsarEnumerateTrustedDomainsEx**  **LsarEnumerateTrustedDomains**  **LsarQueryInfoTrustedDomain**  **LsarSetInformationTrustedDomain**  **LsarEnumeratePrivileges**  **LsarLookupPrivilegeValue**  **LsarLookupPrivilegeName**  **LsarLookupPrivilegeDisplayName**  **LsarQuerySecurityObject**  **LsarSetSecurityObject**  **LsarDeleteObject** |
| [3.1.4.2.3](#Section_5d50b55fe9c74af6bf8502e8043f66ea) | **LsarOpenPolicy2**  **LsarOpenPolicy**  **LsarCreateAccount**  **LsarEnumerateAccounts**  **LsarOpenAccount**  **LsarEnumerateAccountRights**  **LsarRemoveAccountRights**  **LsarCreateSecret**  **LsarOpenSecret**  **LsarRetrievePrivateData** |

##### Access Checks Applied on Handle Open

When opening a handle, the server MUST associate with it a set of ACCESS\_MASK bits, as defined in section [2.2.1.1](#Section_7aeb7f170a6e4f04ac7e7b1363cf9ecf). These access bits control which type of subsequent operations the caller can perform with this handle.

All methods that open handles (as specified in section [3.1.4.1](#Section_879d309fe4d4424b98482913d14994aa)) allow the caller to specify a "desired access" bitmask. The meaning of the bits within this bitmask depends on the type of object. The bits are documented in sections [2.2.1.1.1](#Section_5ee8db785f0e47b2aba78447ff454e3b), [2.2.1.1.2](#Section_b61b7268987a420b84f96c75f8dc8558), [2.2.1.1.3](#Section_fc3b5e24b1a24c7983d7256ceaef8ff4), [2.2.1.1.4](#Section_88c6bd186c404a82ae19fe7bfec5108b), and [2.2.1.1.5](#Section_e035f552031348b79bcafdd9fd4e948e). All methods that open handles (as specified in section 3.1.4.1) perform an access check based on the desired access mask, whose general form is:

1. IF (method specific check fails) THEN
2. Return STATUS\_ACCESS\_DENIED
3. END IF
4. IF (security descriptor check fails) THEN
5. Return STATUS\_ACCESS\_DENIED
6. END IF

The method-specific checks are detailed in the sections for individual methods that open handles. The security-descriptor check is performed by using the Access Check Algorithm Pseudocode ([[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.5.3.2). For this protocol, the input parameters of that algorithm are mapped as follows:

* *SecurityDescriptor*: The [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) of the object to which the handle is being opened, as specified in section [3.1.1](#Section_0877FDC4184F40B0B378F50D6647D23E).
* *Token*: This MUST be the token ([MS-DTYP] section 2.5.2) of the client, obtained by invoking GetRpcImpersonationAccessToken(NULL). The **GetRpcImpersonationAccessToken** interface is specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.3.4.3.1.
* *Access Request mask*: The *DesiredAccess* parameter of the method being invoked, or the *DesiredAccess* value specified in the method description.
* *Object Tree*: This parameter MUST be NULL.
* *PrincipalSelfSubst SID*: This parameter MUST be NULL.
* *GrantedAccess*: The value returned by this parameter MUST be stored in a local variable *LocalGrantedAccess* (of type ACCESS\_MASK).

In the case that the access check is successful, the server MUST NOT grant more access bits than the caller has asked for, but MUST grant only those access bits that the client has explicitly requested. The caller is permitted to request the maximum access permitted by the server by specifying the special constant MAXIMUM\_ALLOWED, as specified in section 2.2.1.1.1.

If *DesiredAccess* contains the MAXIMUM\_ALLOWED bit, the server MUST create and return an LsaContextHandle (section [3.1.1.7](#Section_1011130b0413460d81edd1821d141639)) via the method's LSAPR\_HANDLE\* output parameter, with its fields initialized as follows:

* LsaContextHandle.HandleType = "Policy", "Account", "Secret", or "Trusted Domain", depending on the type of the database object
* LsaContextHandle.Object = the database object
* LsaContextHandle.GrantedAccess = *LocalGrantedAccess*

If *DesiredAccess* does not contain the MAXIMUM\_ALLOWED bit, the following constraint MUST be satisfied:

* If *DesiredAccess* contains bits that are not in *GrantedAccess*, the server MUST return STATUS\_ACCESS\_DENIED. Otherwise, the server MUST create and return an LsaContextHandle (section 3.1.1.7) via the method's LSAPR\_HANDLE\* output parameter, with its fields initialized as follows:
  + LsaContextHandle.HandleType = "Policy", "Account", "Secret", or "Trusted Domain", depending on the type of the database object
  + LsaContextHandle.Object = the database object
  + LsaContextHandle.GrantedAccess = *DesiredAccess*

The server MUST NOT allow the caller to add more access bits to the handle in a subsequent operation. In order to obtain more access, a new handle must be obtained.

##### Access Checks Applied for Object Operations

Each method that consumes a handle requires that certain access bits be set on the handle, which is defined as *RequiredAccess* for the purposes of this specification, and returns STATUS\_ACCESS\_DENIED according to the pseudocode shown below, if necessary. The required bits vary on a per-method basis and might depend on method arguments. Therefore, the value of RequiredAccess is specified on a per-method basis in sections [3.1.4.4](#Section_AB0D94F35F504E44BEB69895A1D0F711), [3.1.4.5](#Section_C6282C425DA94B2C87D914E12E59E786), [3.1.4.6](#Section_C86D4A49E9DD43F88AB144F6BAFFA2A0), [3.1.4.7](#Section_3D7AF62EC260489D8AB5C756E0B9FE57), [3.1.4.8](#Section_1C99AE50D43341778FA656E173831686), and [3.1.4.9](#Section_CF8100EE82D14B6D8CBCA022ADE325BE).

1. IF (any bit set in RequiredAccess is not set in LsaContextHandle.GrantedAccess) THEN
2. Return STATUS\_ACCESS\_DENIED
3. END IF

For example, if a method-processing rule specifies a required access bit of POLICY\_VIEW\_LOCAL\_INFORMATION, the server MUST check that this bit is set in the granted access field on the context handle. If the check is unsuccessful, the server MUST return STATUS\_ACCESS\_DENIED.

##### Determining If Requestors Are Anonymous

1. procedure IsRequestorAnonymous() : boolean

The IsRequestorAnonymous procedure returns TRUE if the requestor is anonymous. On entrance:

* AnonymousSid: This MUST be the ANONYMOUS [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2.4.
* RpcImpersonationAccessToken: This MUST be the token ([MS-DTYP] section 2.5.2) of the client, obtained by invoking the **GetRpcImpersonationAccessToken** interface as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.3.4.3.1, specifying NULL for *Input Parameter*.

1. Return RpcImpersonationAccessToken.Sids[RpcImpersonationAccessToken.UserIndex] equals AnonymousSid

#### Closing Handles

A handle of any type can be closed by calling [LsarClose](#Section_99dd2d7ab0fc4c6d837a2b4d342383ae). Successful calls to [LsarDeleteObject](#Section_8d0aa2dc22b64bc3b5d279b4b0ad7bce), which deletes an object to which the caller has an open handle, MUST also close the handle. The fact that a handle is closed is communicated to the [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a) by returning a NULL value in the handle parameter, as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 5.1.6.

Closing one handle MUST NOT affect any other handle on the server; that is, handles obtained using a policy handle MUST continue to be valid after that policy handle is closed.

#### Policy Object Methods

The message processing of methods in this section MUST use the abstract data model defined in section [3.1.1.1](#Section_c939c70e8cf04d90a9cc1c5002951723).

| Method (opnum) | Summary |
| --- | --- |
| LsarOpenPolicy2 (opnum 44) | Opens a context handle to the [**RPC server**](#gt_ae65dac0-cd24-4e83-a946-6d1097b71553). |
| LsarOpenPolicy (opnum 6) | Superseded by [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b). |
| LsarQueryInformationPolicy2 (opnum 46) | Obtains information from the policy object. |
| LsarQueryInformationPolicy (opnum 7) | Obtains information from the policy object. |
| LsarSetInformationPolicy2 (opnum 47) | Sets information on the policy object. |
| LsarSetInformationPolicy (opnum 8) | Sets information on the policy object. |
| LsarQueryDomainInformationPolicy (opnum 53) | Obtains information from the policy object pertaining to the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). |
| LsarSetDomainInformationPolicy (opnum 54) | Sets information on the policy object pertaining to the domain. |

##### LsarOpenPolicy2 (Opnum 44)

The LsarOpenPolicy2 method opens a context handle to the [**RPC server**](#gt_ae65dac0-cd24-4e83-a946-6d1097b71553). This is the first function that MUST be called to contact the Local Security Authority (Domain Policy) Remote Protocol database.

1. NTSTATUS LsarOpenPolicy2(
2. [in, unique, string] wchar\_t\* SystemName,
3. [in] PLSAPR\_OBJECT\_ATTRIBUTES ObjectAttributes,
4. [in] ACCESS\_MASK DesiredAccess,
5. [out] LSAPR\_HANDLE\* PolicyHandle
6. );

**SystemName:** This parameter does not have any effect on message processing in any environment. It MUST be ignored on receipt.

**ObjectAttributes:** This parameter does not have any effect on message processing in any environment. All fields MUST[<54>](#Appendix_A_54" \o "Product behavior note 54) be ignored except **RootDirectory** which MUST be NULL.

**DesiredAccess:** An [ACCESS\_MASK](#Section_7aeb7f170a6e4f04ac7e7b1363cf9ecf) value that specifies the requested access rights that MUST be granted on the returned PolicyHandle if the request is successful.

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle (as specified in section [2.2.2.1](#Section_0d093105e8c845f7a79d182aafd60c6e)) that represents a reference to the abstract data model of a policy object, as specified in section [3.1.1.1](#Section_c939c70e8cf04d90a9cc1c5002951723).

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing below.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied parameters is incorrect. For example, this can happen when *ObjectAttributes* is NULL or *DesiredAccess* is zero. |

Processing:

*DesiredAccess*: A bitmask specifying the access that the caller attempts to obtain on the policy object, which is access-checked according to section [3.1.4.2.1](#Section_e5e1e32e4066435db669044fe997eaf7). The method-specific portion of the check is the following:

1. LET serverInfo be a SERVER\_INFO\_101 structure
2. CALL ServerGetInfo(101, &serverInfo)
3. LET isDomainController be a boolean initialized to FALSE
4. IF (serverInfo.sv101\_version\_type & (SV\_TYPE\_DOMAIN\_CTRL | SV\_TYPE\_DOMAIN\_BAKCTRL)) THEN
5. Set isDomainController equal to TRUE
6. END IF
8. IF ((isDomainController equals FALSE) and (IsRequestorAnonymous() and LsaRestrictAnonymous is set to TRUE)) THEN
9. Return STATUS\_ACCESS\_DENIED
10. END IF

SERVER\_INFO\_101, SV\_TYPE\_DOMAIN\_CTRL, and SV\_TYPE\_DOMAIN\_BACKCTRL are specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.3.12. The ServerGetInfo procedure is specified in [MS-DTYP] section 2.6. The valid account-rights bits are specified in section [2.2.1.1.2](#Section_B61B7268987A420B84F96C75F8DC8558), and the [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) is specified in section 3.1.1.1. The IsRequestorAnonymous procedure is specified in section [3.1.4.2.3](#Section_5d50b55fe9c74af6bf8502e8043f66ea).

*PolicyHandle*: If the request is successful, the server MUST create and return a context handle (section [3.1.1.7](#Section_1011130b0413460d81edd1821d141639)) via *PolicyHandle*, with its fields initialized as follows:

* LsaContextHandle.HandleType = "Policy"
* LsaContextHandle.Object = the policy object
* LsaContextHandle.GrantedAccess = as specified in section 3.1.4.2.1

The return value MUST be set to STATUS\_SUCCESS in this case.

##### LsarOpenPolicy (Opnum 6)

The LsarOpenPolicy method is exactly the same as [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b), except that the *SystemName* parameter in this function, because of its syntactic definition, contains only one character instead of a full string. This *SystemName* parameter does not have any effect on message processing in any environment. It MUST be ignored.

1. NTSTATUS LsarOpenPolicy(
2. [in, unique] wchar\_t\* SystemName,
3. [in] PLSAPR\_OBJECT\_ATTRIBUTES ObjectAttributes,
4. [in] ACCESS\_MASK DesiredAccess,
5. [out] LSAPR\_HANDLE\* PolicyHandle
6. );

**SystemName:** This parameter does not have any effect on message processing in any environment. It MUST be ignored on receipt.

**ObjectAttributes:** This parameter does not have any effect on message processing in any environment. All fields MUST[<55>](#Appendix_A_55" \o "Product behavior note 55) be ignored except **RootDirectory**, which MUST be NULL.

**DesiredAccess:** An [ACCESS\_MASK](#Section_7aeb7f170a6e4f04ac7e7b1363cf9ecf) value that specifies the requested access rights that MUST be granted on the returned PolicyHandle, if the request is successful.

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle (as specified in section [2.2.2.1](#Section_0d093105e8c845f7a79d182aafd60c6e)) that represents a reference to the abstract data model of a policy object, as specified in section [3.1.1.1](#Section_c939c70e8cf04d90a9cc1c5002951723).

Processing:

The processing is the same as for LsarOpenPolicy2. LsarOpenPolicy2 supersedes this message and MUST be used when possible.

##### LsarQueryInformationPolicy2 (Opnum 46)

The LsarQueryInformationPolicy2 method is invoked to query values that represent the server's security policy.

1. NTSTATUS LsarQueryInformationPolicy2(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] POLICY\_INFORMATION\_CLASS InformationClass,
4. [out, switch\_is(InformationClass)]
5. PLSAPR\_POLICY\_INFORMATION\* PolicyInformation
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**InformationClass:** A parameter that specifies what type of information the caller is requesting.

**PolicyInformation:** A parameter that references policy information structure on return.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing below.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC000009A  STATUS\_INSUFFICIENT\_RESOURCES | There are insufficient resources to complete the request. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform the operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the parameters is incorrect. For instance, this can happen if *InformationClass* is out of range or if *PolicyInformation* is NULL. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

*PolicyHandle* MUST be a handle to an open policy object, and *PolicyHandle*.HandleType MUST equal "Policy"; otherwise, STATUS\_INVALID\_HANDLE MUST be returned.

The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03). The following table specifies the RequiredAccess value to use in this access check for each *InformationClass* value or indicates if no processing is supported, regardless of access granted.

| InformationClass value | RequiredAccess value |
| --- | --- |
| PolicyAuditLogInformation | POLICY\_VIEW\_AUDIT\_INFORMATION |
| PolicyAuditEventsInformation | POLICY\_VIEW\_AUDIT\_INFORMATION |
| PolicyPrimaryDomainInformation | POLICY\_VIEW\_LOCAL\_INFORMATION |
| PolicyPdAccountInformation | POLICY\_GET\_PRIVATE\_INFORMATION |
| PolicyAccountDomainInformation | POLICY\_VIEW\_LOCAL\_INFORMATION |
| PolicyLsaServerRoleInformation | POLICY\_VIEW\_LOCAL\_INFORMATION |
| PolicyReplicaSourceInformation | POLICY\_VIEW\_LOCAL\_INFORMATION |
| PolicyModificationInformation | Not applicable: This information class cannot be queried. The request MUST fail with STATUS\_INVALID\_PARAMETER. |
| PolicyAuditFullSetInformation | Not applicable: This information class cannot be queried. The request MUST fail with STATUS\_INVALID\_PARAMETER. |
| PolicyAuditFullQueryInformation | POLICY\_VIEW\_AUDIT\_INFORMATION |
| PolicyDnsDomainInformation | POLICY\_VIEW\_LOCAL\_INFORMATION |
| PolicyDnsDomainInformationInt | POLICY\_VIEW\_LOCAL\_INFORMATION |
| PolicyLocalAccountDomainInformation | POLICY\_VIEW\_LOCAL\_INFORMATION |

The *InformationClass* parameter can take on any value in the POLICY\_INFORMATION\_CLASS enumeration range. For all values outside this range, the server MUST return a STATUS\_INVALID\_PARAMETER error code.

*PolicyInformation* is an output parameter. The server MUST fill it in with the information requested by the client, based on the value of the *InformationClass* parameter and the abstract data model specified in section [3.1.1.1](#Section_c939c70e8cf04d90a9cc1c5002951723), as follows.

| Value of InformationClass parameter | Information returned to caller from abstract data model |
| --- | --- |
| PolicyAuditLogInformation | Auditing Log Information |
| PolicyAuditEventsInformation | Event Auditing Options |
| PolicyPrimaryDomainInformation | Primary Domain Information |
| PolicyPdAccountInformation | MUST return an [LSAPR\_POLICY\_PD\_ACCOUNT\_INFO](#Section_b04175b3fedf4dda9034f754a10fe64e) information structure, its **Name** member being an RPC\_UNICODE\_STRING with **Length** set to 0 and **Buffer** initialized to NULL. |
| PolicyAccountDomainInformation | On non–[**domain controllers**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd): Account Domain  On domain controller: Primary Domain Information |
| PolicyLsaServerRoleInformation | Server Role Information |
| PolicyReplicaSourceInformation | Replica Source Information |
| PolicyModificationInformation | MUST return STATUS\_INVALID\_PARAMETER |
| PolicyAuditFullSetInformation | MUST return STATUS\_INVALID\_PARAMETER |
| PolicyAuditFullQueryInformation | Audit Full Information[<56>](#Appendix_A_56" \o "Product behavior note 56) |
| PolicyDnsDomainInformation | DNS Domain Information[<57>](#Appendix_A_57" \o "Product behavior note 57) |
| PolicyDnsDomainInformationInt | DNS Domain Information |
| PolicyLocalAccountDomainInformation | AccountDomainInformation |

##### LsarQueryInformationPolicy (Opnum 7)

The LsarQueryInformationPolicy method is invoked to query values that represent the server's information policy.

1. NTSTATUS LsarQueryInformationPolicy(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] POLICY\_INFORMATION\_CLASS InformationClass,
4. [out, switch\_is(InformationClass)]
5. PLSAPR\_POLICY\_INFORMATION\* PolicyInformation
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**InformationClass:** A parameter that specifies what type of information the caller is requesting.

**PolicyInformation:** A parameter that references policy information structure on return.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing below.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC000009A  STATUS\_INSUFFICIENT\_RESOURCES | There are insufficient resources to complete the request. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform the operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the parameters is incorrect. For instance, this can happen if *InformationClass* is out of range or if *PolicyInformation* is NULL. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message MUST be processed in an identical manner to [LsarQueryInformationPolicy2](#Section_516F503C0230489DB012E650B46B66A2).

##### LsarSetInformationPolicy2 (Opnum 47)

The LsarSetInformationPolicy2 method is invoked to set a policy on the server.

1. NTSTATUS LsarSetInformationPolicy2(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] POLICY\_INFORMATION\_CLASS InformationClass,
4. [in, switch\_is(InformationClass)]
5. PLSAPR\_POLICY\_INFORMATION PolicyInformation
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**InformationClass:** A parameter that specifies what type of information the caller is setting.

**PolicyInformation:** Data that represents policy being set.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the parameters is incorrect. For instance, this can happen if *InformationClass* is not supported or some of the supplied policy data is invalid. |
| 0xC0000002  STATUS\_NOT\_IMPLEMENTED | This information class cannot be set. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

*PolicyHandle* MUST reference a context that was granted an access commensurate with the *InformationClass* value requested. If *PolicyHandle* is not a valid context handle or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. If the context does not have sufficient access, the server MUST return STATUS\_ACCESS\_DENIED.

The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03). The following table specifies the RequiredAccess value to use in this access check for each *InformationClass* value or indicates if no processing is supported, regardless of access granted.

| InformationClass value | RequiredAccess value |
| --- | --- |
| PolicyAuditLogInformation | POLICY\_AUDIT\_LOG\_ADMIN |
| PolicyAuditEventsInformation | POLICY\_SET\_AUDIT\_REQUIREMENTS |
| PolicyPrimaryDomainInformation | POLICY\_TRUST\_ADMIN |
| PolicyPdAccountInformation | Not applicable: This information class cannot be set; the request MUST fail with STATUS\_INVALID\_PARAMETER. |
| PolicyAccountDomainInformation | Not applicable: This information class cannot be set; the request MUST fail with STATUS\_INVALID\_PARAMETER. |
| PolicyLsaServerRoleInformation | POLICY\_SERVER\_ADMIN |
| PolicyReplicaSourceInformation | POLICY\_SERVER\_ADMIN |
| PolicyModificationInformation | Not applicable: This information class cannot be set; the request MUST fail with STATUS\_INVALID\_PARAMETER. |
| PolicyAuditFullSetInformation | Not applicable: This information class cannot be set; the request MUST fail with STATUS\_INVALID\_PARAMETER. |
| PolicyAuditFullQueryInformation | Not applicable: This information class cannot be set; the request MUST fail with STATUS\_INVALID\_PARAMETER. |
| PolicyDnsDomainInformation | POLICY\_TRUST\_ADMIN |
| PolicyDnsDomainInformationInt | POLICY\_TRUST\_ADMIN |
| PolicyLocalAccountDomainInformation | POLICY\_TRUST\_ADMIN |

The *InformationClass* parameter can take on any value in the POLICY\_INFORMATION\_CLASS enumeration range. For all values outside this range, the server MUST return the STATUS\_INVALID\_PARAMETER error code.

The *PolicyInformation* parameter contains the data that the caller wishes to set, based on the value of the *InformationClass* parameter. The server MUST update its abstract data model, specified in section [3.1.1.1](#Section_c939c70e8cf04d90a9cc1c5002951723), as follows.

| Value of InformationClass parameter | Information updated in abstract data model |
| --- | --- |
| PolicyAuditLogInformation | Server MUST return the STATUS\_NOT\_IMPLEMENTED error code because this is not a policy element that can be set. |
| PolicyAuditEventsInformation | Event Auditing Options. |
| PolicyPrimaryDomainInformation | Primary Domain Information. |
| PolicyPdAccountInformation | Server MUST return STATUS\_INVALID\_PARAMETER because this is not a policy element that can be set. |
| PolicyAccountDomainInformation | On a [**domain controller**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd), the server MUST fail this request with the STATUS\_INVALID\_PARAMETER.  On non-domain controllers: Account Domain Information. |
| PolicyLsaServerRoleInformation | Server Role Information. |
| PolicyReplicaSourceInformation | Replica Source Information. |
| PolicyModificationInformation | Server MUST return STATUS\_INVALID\_PARAMETER because this is not a policy element that can be set. |
| PolicyAuditFullSetInformation | **ShutDownOnFull** field of Audit Full Information.[<58>](#Appendix_A_58" \o "Product behavior note 58) |
| PolicyAuditFullQueryInformation | Server MUST record STATUS\_INVALID\_PARAMETER because this is not a policy element that can be set. |
| PolicyDnsDomainInformation | DNS Domain Information.[<59>](#Appendix_A_59" \o "Product behavior note 59) |
| PolicyDnsDomainInformationInt | DNS Domain Information. |
| PolicyLocalAccountDomainInformation | Account Domain Information. |

##### LsarSetInformationPolicy (Opnum 8)

The LsarSetInformationPolicy method is invoked to set a policy on the server.

1. NTSTATUS LsarSetInformationPolicy(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] POLICY\_INFORMATION\_CLASS InformationClass,
4. [in, switch\_is(InformationClass)]
5. PLSAPR\_POLICY\_INFORMATION PolicyInformation
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**InformationClass:** A parameter that specifies what type of information the caller is setting.

**PolicyInformation:** Data that represents the policy being set.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the parameters is incorrect. For instance, this can happen if *InformationClass* is not supported or some of the supplied policy data is invalid. |
| 0xC0000002  STATUS\_NOT\_IMPLEMENTED | This information class cannot be set. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message MUST be processed in an identical manner to [LsarSetInformationPolicy2](#Section_fc14e9aea26e4031809ea908dd3e13a3).

##### LsarQueryDomainInformationPolicy (Opnum 53)

The LsarQueryDomainInformationPolicy method is invoked to retrieve policy settings in addition to those exposed through [LsarQueryInformationPolicy](#Section_3564ba7084ea4f04a9dcdede9f96a8bf) and [LsarSetInformationPolicy2](#Section_fc14e9aea26e4031809ea908dd3e13a3). Despite the term "Domain" in the name of the method, processing of this message occurs with local data, and furthermore, there is no requirement that this data have any relationship with the LSA information in the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) to which the machine is joined.

1. NTSTATUS LsarQueryDomainInformationPolicy(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] POLICY\_DOMAIN\_INFORMATION\_CLASS InformationClass,
4. [out, switch\_is(InformationClass)]
5. PLSAPR\_POLICY\_DOMAIN\_INFORMATION\* PolicyDomainInformation
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**InformationClass:** A parameter that specifies what type of information the caller is requesting.

**PolicyDomainInformation:** A parameter that references policy information structure on return.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied arguments was invalid. |
| 0xC0000034  STATUS\_OBJECT\_NAME\_NOT\_FOUND | No value has been set for this policy. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

If the *InformationClass* parameter is PolicyDomainEfsInformation, and the responder does not support Encrypting File System (EFS) Policy Information as specified in section [3.1.1.1](#Section_c939c70e8cf04d90a9cc1c5002951723), the request MUST fail with STATUS\_OBJECT\_NAME\_NOT\_FOUND.

If the *InformationClass* parameter is PolicyDomainQualityOfServiceInformation, and the responder implementation does not support Quality Of Service Information as specified in section 3.1.1.1, the request MUST fail with STATUS\_INVALID\_PARAMETER.

If *PolicyHandle* is not a valid context handle or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE.

The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03). The following table specifies the RequiredAccess value to use in this access check for each *InformationClass* value.

| InformationClass value | RequiredAccess value |
| --- | --- |
| PolicyDomainQualityOfServiceInformation | POLICY\_VIEW\_AUDIT\_INFORMATION |
| PolicyDomainEfsInformation | POLICY\_VIEW\_LOCAL\_INFORMATION |
| PolicyDomainKerberosTicketInformation | POLICY\_VIEW\_LOCAL\_INFORMATION |

The *InformationClass* parameter can take on any value in the POLICY\_DOMAIN\_INFORMATION\_CLASS enumeration range. For all values outside this range, the server MUST return the STATUS\_INVALID\_PARAMETER error code.

*PolicyDomainInformation* is an output parameter. The server MUST fill it with the information requested by the client, based on the value of the *InformationClass* parameter and the abstract data model specified in section 3.1.1.1. If the information has not been set before, the request MUST fail with STATUS\_OBJECT\_NAME\_NOT\_FOUND.

| Value of InformationClass parameter | Information returned to caller from abstract data model |
| --- | --- |
| PolicyDomainQualityOfServiceInformation | Quality Of Service Information |
| PolicyDomainEfsInformation | EFS Policy Information |
| PolicyDomainKerberosTicketInformation | Kerberos Policy Information |

##### LsarSetDomainInformationPolicy (Opnum 54)

The LsarSetDomainInformationPolicy method is invoked to change policy settings in addition to those exposed through [LsarQueryInformationPolicy](#Section_3564ba7084ea4f04a9dcdede9f96a8bf) and [LsarSetInformationPolicy2](#Section_fc14e9aea26e4031809ea908dd3e13a3). Despite the term "Domain" in the name of the method, processing of this message occurs with local data. Also, there is no requirement that this data have any relationship with the LSA information in the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) in which the machine is joined.

1. NTSTATUS LsarSetDomainInformationPolicy(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] POLICY\_DOMAIN\_INFORMATION\_CLASS InformationClass,
4. [in, unique, switch\_is(InformationClass)]
5. PLSAPR\_POLICY\_DOMAIN\_INFORMATION PolicyDomainInformation
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**InformationClass:** A parameter that specifies what type of information the caller is setting.

**PolicyDomainInformation:** Data representing policy being set.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the following message processing.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied arguments was invalid. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

If the *InformationClass* parameter is PolicyDomainEfsInformation, and the responder implementation does not support Encrypting File System (EFS) Policy Information as specified in section [3.1.1.1](#Section_c939c70e8cf04d90a9cc1c5002951723), the request MUST fail with STATUS\_INVALID\_PARAMETER.

If the *InformationClass* parameter is PolicyDomainQualityOfServiceInformation, and the responder implementation does not support Quality Of Service Information as specified in section 3.1.1.1, the request MUST fail with an RPC exception RPC\_S\_INVALID\_TAG.

If *PolicyHandle* is not a valid context handle or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE.

The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03). The following table specifies the RequiredAccess value to use in this access check for each *InformationClass* value.

| InformationClass value | RequiredAccess value |
| --- | --- |
| PolicyDomainQualityOfServiceInformation | POLICY\_SERVER\_ADMIN |
| PolicyDomainEfsInformation | POLICY\_SERVER\_ADMIN |
| PolicyDomainKerberosTicketInformation | POLICY\_SERVER\_ADMIN |

The *InformationClass* parameter can take on any value in the POLICY\_DOMAIN\_INFORMATION\_CLASS enumeration range. For all values outside this range, the server MUST return the STATUS\_INVALID\_PARAMETER error code.

The *PolicyDomainInformation* parameter contains the data that the caller needs to set, based on the value of the *InformationClass* parameter. The server MUST update its abstract data model, specified in section 3.1.1.1, as follows.

| Value of InformationClass parameter | Information returned to caller from abstract data model |
| --- | --- |
| PolicyDomainQualityOfServiceInformation | Quality Of Service Information |
| PolicyDomainEfsInformation | EFS Policy Information |
| PolicyDomainKerberosTicketInformation | Kerberos Policy Information |

If the abstract data model update succeeds and the *InformationClass* parameter is PolicyDomainKerberosTicketInformation, the server MUST invoke the KDC ConfigurationChange event (see [[MS-KILE]](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9) section 3.3.4.1) and MUST ignore any errors that are returned.

#### Account Object Methods

The message processing of methods in this section MUST use the abstract data model, as specified in section [3.1.1.3](#Section_d1a82c1b4d2b456fb4dfc70b461f3d42).

| Method (opnum) | Summary |
| --- | --- |
| LsarCreateAccount (opnum 10) | Creates a new [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b) in the policy database. |
| LsarEnumerateAccounts (opnum 11) | Enumerates all account objects in the policy database. |
| LsarOpenAccount (opnum 17) | Opens a handle to an existing account object. |
| LsarEnumeratePrivilegesAccount (opnum 18) | Enumerates all rights and [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) of an account. |
| LsarAddPrivilegesToAccount (opnum 19) | Adds new privileges to an existing account object. |
| LsarRemovePrivilegesFromAccount (opnum 20) | Removes privileges from an existing account object. |
| LsarGetSystemAccessAccount (opnum 23) | Retrieves system access flags from the account object. |
| LsarSetSystemAccessAccount (opnum 24) | Sets system access flags on the account object. |
| LsarEnumerateAccountsWithUserRight (opnum 35) | Enumerates all account objects in the server's policy database that match a given user right. |
| LsarEnumerateAccountRights (opnum 36) | Enumerates all rights of an account object in the server's policy database. |
| LsarAddAccountRights (opnum 37) | Adds new rights to an account object in the server's policy database. |
| LsarRemoveAccountRights (opnum 38) | Removes rights from an account object in the server's policy database. |

##### LsarCreateAccount (Opnum 10)

The LsarCreateAccount method is invoked to create a new [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b) in the server's database.

1. NTSTATUS LsarCreateAccount(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_SID AccountSid,
4. [in] ACCESS\_MASK DesiredAccess,
5. [out] LSAPR\_HANDLE\* AccountHandle
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**AccountSid:** The [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the account to be created.

**DesiredAccess:** A bitmask specifying accesses to be granted to the newly created and opened account at this time.

**AccountHandle:** Used to return a handle to the newly created account object.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC0000035  STATUS\_OBJECT\_NAME\_COLLISION | An account with this SID already exists. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | Some of the parameters supplied were invalid. |

Processing:

This message takes four arguments:

*PolicyHandle*: A handle to an open policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to POLICY\_CREATE\_ACCOUNT.

*AccountSid*: The SID of the account to be created. The server MUST validate that *AccountSid* represents a valid SID and fail the request with STATUS\_INVALID\_PARAMETER if it is not.[<60>](#Appendix_A_60" \o "Product behavior note 60)

*DesiredAccess*: A set of access bits that the caller attempts to receive from the account object after it has been created, which is access-checked according to section 3.1.4.2.2. The method-specific portion of the check is the following.

1. IF (IsRequestorAnonymous() and LsaRestrictAnonymous is set to TRUE) THEN
2. Return STATUS\_OBJECT\_NAME\_NOT\_FOUND
3. END IF

The valid account-rights bits are specified in section [2.2.1.1.3](#Section_FC3B5E24B1A24C7983D7256CEAEF8FF4), and the [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) is specified in section [3.1.1.3](#Section_D1A82C1B4D2B456FB4DFC70B461F3D42). The IsRequestorAnonymous procedure is specified in section [3.1.4.2.3](#Section_5d50b55fe9c74af6bf8502e8043f66ea).

*AccountHandle*: If the request is successful, the server MUST create and return a context handle (section [3.1.1.7](#Section_1011130b0413460d81edd1821d141639)) via *AccountHandle*, with its fields initialized as follows:

* LsaContextHandle.HandleType = "Account"
* LsaContextHandle.Object = the account object
* LsaContextHandle.GrantedAccess = as specified in section [3.1.4.2.1](#Section_e5e1e32e4066435db669044fe997eaf7)

This mechanism allows the caller to skip the additional step of opening the account object after creating it.

The server MUST check whether another account object already exists in its policy database with the same SID, and fail the request with STATUS\_OBJECT\_NAME\_COLLISION if it does.

The server MUST associate a security descriptor with a newly created account object. See section 3.1.1.3 for the data model of this object type.

##### LsarEnumerateAccounts (Opnum 11)

The LsarEnumerateAccounts method is invoked to request a list of [**account objects**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b) in the server's database. The method can be called multiple times to return its output in fragments.

1. NTSTATUS LsarEnumerateAccounts(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] [out] unsigned long \*EnumerationContext,
4. [out] PLSAPR\_ACCOUNT\_ENUM\_BUFFER EnumerationBuffer,
5. [in] unsigned long PreferedMaximumLength
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**EnumerationContext:** A pointer to a context value that is used to resume enumeration, if necessary.

**EnumerationBuffer:** A pointer to a structure that will contain the results of the enumeration.

**PreferedMaximumLength:** A value that indicates the approximate size of the data to return.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0x00000105  STATUS\_MORE\_ENTRIES | More information is available to successive calls. |
| 0x8000001A  STATUS\_NO\_MORE\_ENTRIES | No more entries are available from the enumeration. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message takes four arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to POLICY\_VIEW\_LOCAL\_INFORMATION. If IsRequestorAnonymous() returns TRUE (section [3.1.4.2.3](#Section_5d50b55fe9c74af6bf8502e8043f66ea)) and LsaRestrictAnonymous is set to TRUE, the call MUST fail with STATUS\_ACCESS\_DENIED.

*EnumerationContext*: A number that indicates a starting index at which to begin the enumeration. The server MUST always return all account objects in the same order, starting at the object whose index is *EnumerationContext*. To initiate a new enumeration, the client sets *EnumerationContext* to zero; otherwise, the client sets *EnumerationContext* to a value returned by a previous call to the method.

The server MUST return STATUS\_INVALID\_PARAMETER if the *EnumerationContext* parameter is NULL.

*EnumerationBuffer*: Used to return the results of enumeration. The server MUST fill *EnumerationBuffer* with as many account objects as possible, as determined by *PreferedMaximumLength*. If the size of all remaining objects is less than or equal to *PreferedMaximumLength*, the server MUST fill *EnumerationBuffer* with all objects. If the size of all remaining objects is greater than *PreferedMaximumLength*, the server MUST fill *EnumerationBuffer* with objects such that the size of the account objects returned is greater than or equal to *PreferedMaximumLength*, but would be less than *PreferedMaximumLength* if the last object had not been added to *EnumerationBuffer*. If there are no more objects than are returned in *EnumerationBuffer*, the server MUST return STATUS\_NO\_MORE\_ENTRIES. If there are more database objects than are returned in *EnumerationBuffer*, the server MUST set the *EnumerationContext* value to the index value that would allow it to resume enumeration correctly when this method is called again, and the server MUST return STATUS\_MORE\_ENTRIES. Note that this return value is not an error status.

*PreferedMaximumLength*: An indication about the approximate size, in bytes, of the data to return. Any unsigned 32-bit value is valid for the *PreferedMaximumLength* parameter.

##### LsarOpenAccount (Opnum 17)

The LsarOpenAccount method is invoked to obtain a handle to an [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b).

1. NTSTATUS LsarOpenAccount(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_SID AccountSid,
4. [in] ACCESS\_MASK DesiredAccess,
5. [out] LSAPR\_HANDLE\* AccountHandle
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**AccountSid:** A [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the account to be opened.

**DesiredAccess:** A bitmask specifying accesses to be granted to the opened account at this time.

**AccountHandle:** Used to return a handle to the opened account object.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | Some of the parameters supplied are incorrect. For instance, this can happen when *AccountSid* is NULL. |
| 0xC0000034  STATUS\_OBJECT\_NAME\_NOT\_FOUND | An account with this SID does not exist in the server's database. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message takes four arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. *PolicyHandle*.GrantedAccess MUST NOT be considered for this call, because the access check MUST happen on the account object.

*AccountSid*: The SID of the account to be opened. The server MUST verify that the SID is valid and fail the request with STATUS\_INVALID\_PARAMETER otherwise. The server MUST verify that the account object with this SID exists in its policy database and fail the request with STATUS\_OBJECT\_NAME\_NOT\_FOUND otherwise.

*DesiredAccess*: A bitmask specifying the type of access the caller attempts to obtain from the account object, which is access-checked according to section [3.1.4.2.1](#Section_e5e1e32e4066435db669044fe997eaf7). The method-specific portion of the check is the following.

1. IF (IsRequestorAnonymous() and LsaRestrictAnonymous is set to TRUE) THEN
2. Return STATUS\_OBJECT\_NAME\_NOT\_FOUND
3. END IF

The valid account rights bits are specified in section [2.2.1.1.3](#Section_fc3b5e24b1a24c7983d7256ceaef8ff4), and the [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) is specified in section [3.1.1.3](#Section_D1A82C1B4D2B456FB4DFC70B461F3D42). The IsRequestorAnonymous procedure is specified in section [3.1.4.2.3](#Section_5d50b55fe9c74af6bf8502e8043f66ea).

*AccountHandle*: If the request is successful, this parameter is used to return a handle (section [3.1.1.7](#Section_1011130b0413460d81edd1821d141639)) to the opened account object with its fields initialized as follows:

* LsaContextHandle.HandleType = "Account"
* LsaContextHandle.Object = the account object
* LsaContextHandle.GrantedAccess = as specified in section 3.1.4.2.1

##### LsarEnumeratePrivilegesAccount (Opnum 18)

The LsarEnumeratePrivilegesAccount method is invoked to retrieve a list of [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) granted to an account on the server.

1. NTSTATUS LsarEnumeratePrivilegesAccount(
2. [in] LSAPR\_HANDLE AccountHandle,
3. [out] PLSAPR\_PRIVILEGE\_SET\* Privileges
4. );

**AccountHandle:** An open [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b) handle obtained from either [LsarCreateAccount (section 3.1.4.5.1)](#Section_841e32115be44b509f112d4941c40a30) or [LsarOpenAccount (section 3.1.4.5.3)](#Section_355e2952abe447c396d9a2f4bd01cf3d).

**Privileges:** Used to return a list of privileges granted to the account.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC000009A  STATUS\_INSUFFICIENT\_RESOURCES | There are insufficient resources to complete the request. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *AccountHandle* is not a valid handle. |

Processing:

This message takes two arguments:

*AccountHandle*: An open handle to an account object. If the handle is not a valid context handle to an account object or *AccountHandle*.HandleType does not equal "Account", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *AccountHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to ACCOUNT\_VIEW.

*Privileges*: Used to return a set of privileges associated with the account. It is valid for the set of privileges to be empty.

The server MUST return STATUS\_INSUFFICIENT\_RESOURCES if it runs out of memory while processing this request.

##### LsarAddPrivilegesToAccount (Opnum 19)

The LsarAddPrivilegesToAccount method is invoked to add new [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) to an existing [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b).

1. NTSTATUS LsarAddPrivilegesToAccount(
2. [in] LSAPR\_HANDLE AccountHandle,
3. [in] PLSAPR\_PRIVILEGE\_SET Privileges
4. );

**AccountHandle:** An open account object handle obtained from either [LsarCreateAccount (section 3.1.4.5.1)](#Section_841e32115be44b509f112d4941c40a30) or [LsarOpenAccount (section 3.1.4.5.3)](#Section_355e2952abe447c396d9a2f4bd01cf3d).

**Privileges:** Contains a list of privileges to add to the account.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | Some of the parameters supplied were invalid. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *AccountHandle* is not a valid handle. |

Processing:

This message takes two arguments:

*AccountHandle*: An open handle to an account object. If the handle is not a valid context handle to an account object or *AccountHandle*.HandleType does not equal "Account", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *AccountHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to ACCOUNT\_ADJUST\_PRIVILEGES.

*Privileges*: A set of privileges to add to an account. Each privilege is a [**LUID**](#gt_96b64af9-1896-4bde-b988-54d469c5affd)-Attributes pair where the **Luid** field MUST match a LUID of a privilege on the server. The attributes replace any attributes of the privilege if one was associated with the account previously. Any LUID not recognized as valid by the server SHOULD cause the message to be rejected with STATUS\_INVALID\_PARAMETER.[<61>](#Appendix_A_61" \o "Product behavior note 61)

##### LsarRemovePrivilegesFromAccount (Opnum 20)

The LsarRemovePrivilegesFromAccount method is invoked to remove [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) from an [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b).

1. NTSTATUS LsarRemovePrivilegesFromAccount(
2. [in] LSAPR\_HANDLE AccountHandle,
3. [in] unsigned char AllPrivileges,
4. [in, unique] PLSAPR\_PRIVILEGE\_SET Privileges
5. );

**AccountHandle:** An open account object handle obtained from either [LsarCreateAccount (section 3.1.4.5.1)](#Section_841e32115be44b509f112d4941c40a30) or [LsarOpenAccount (section 3.1.4.5.3)](#Section_355e2952abe447c396d9a2f4bd01cf3d).

**AllPrivileges:** If this parameter is not FALSE (0), all privileges will be stripped from the account object.

**Privileges:** Contains a (possibly empty) list of privileges to remove from the account object.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | Some of the parameters supplied were invalid. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *AccountHandle* is not a valid handle. |

Processing:

This message takes three arguments:

*AccountHandle*: An open handle to an account object. If the handle is not a valid context handle to an account object or *AccountHandle*.HandleType does not equal "Account", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *AccountHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to ACCOUNT\_ADJUST\_PRIVILEGES.

*AllPrivileges*: A Boolean value; if not FALSE (0), all privileges associated with the account are removed. In this case, the server MUST check that the *Privileges* parameter is NULL, and fail the request with STATUS\_INVALID\_PARAMETER otherwise.

*Privileges*: If *AllPrivileges* is FALSE (0), this parameter cannot be NULL. It will be used to remove *Privileges* from the account object. The server MUST verify that *Privileges* is not NULL and fail the request with STATUS\_INVALID\_PARAMETER otherwise.[<62>](#Appendix_A_62" \o "Product behavior note 62)

##### LsarGetSystemAccessAccount (Opnum 23)

The LsarGetSystemAccessAccount method is invoked to retrieve system access account flags for an [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b). System access account flags are described as part of the account object data model, as specified in section [3.1.1.3](#Section_d1a82c1b4d2b456fb4dfc70b461f3d42).

1. NTSTATUS LsarGetSystemAccessAccount(
2. [in] LSAPR\_HANDLE AccountHandle,
3. [out] unsigned long\* SystemAccess
4. );

**AccountHandle:** An open account object handle obtained from either [LsarCreateAccount (section 3.1.4.5.1)](#Section_841e32115be44b509f112d4941c40a30) or [LsarOpenAccount (section 3.1.4.5.3)](#Section_355e2952abe447c396d9a2f4bd01cf3d).

**SystemAccess:** Used to return a bitmask of access flags associated with the account.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *AccountHandle* is not a valid handle. |

Processing:

This message takes two arguments:

*AccountHandle*: An open handle to an account object. If the handle is not a valid context handle to an account object or *AccountHandle*.HandleType does not equal "Account", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *AccountHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to ACCOUNT\_VIEW.

*SystemAccess*: Used to return a bitmask of system access bits.

##### LsarSetSystemAccessAccount (Opnum 24)

The LsarSetSystemAccessAccount method is invoked to set system access account flags for an [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b).

1. NTSTATUS LsarSetSystemAccessAccount(
2. [in] LSAPR\_HANDLE AccountHandle,
3. [in] unsigned long SystemAccess
4. );

**AccountHandle:** An open account object handle obtained from either [LsarCreateAccount (section 3.1.4.5.1)](#Section_841e32115be44b509f112d4941c40a30) or [LsarOpenAccount (section 3.1.4.5.3)](#Section_355e2952abe447c396d9a2f4bd01cf3d).

**SystemAccess:** A bitmask containing the account flags to be set on the account.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied parameters was invalid. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *AccountHandle* is not a valid handle. |

Processing:

This message takes two arguments:

*AccountHandle*: An open handle to an account object. If the handle is not a valid context handle to an account object or *AccountHandle*.HandleType does not equal "Account", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that AccountHandle grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to ACCOUNT\_ADJUST\_SYSTEM\_ACCESS.

*SystemAccess*: Specifies the set of access bits to be added to account's system access. The server MUST verify that the bits do not fall outside the set of system access rights defined on the system, and fail the request with STATUS\_INVALID\_PARAMETER otherwise. The new system access bits replace the old ones.

##### LsarEnumerateAccountsWithUserRight (Opnum 35)

The LsarEnumerateAccountsWithUserRight method is invoked to return a list of [**account objects**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b) that have the user right equal to the passed-in value.

1. NTSTATUS LsarEnumerateAccountsWithUserRight(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in, unique] PRPC\_UNICODE\_STRING UserRight,
4. [out] PLSAPR\_ACCOUNT\_ENUM\_BUFFER EnumerationBuffer
5. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**UserRight:** The name of the right to use in enumeration.

**EnumerationBuffer:** Used to return the list of account objects that have the specified right.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC0000060  STATUS\_NO\_SUCH\_PRIVILEGE | The supplied name is not recognized by the server. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied arguments is invalid. |
| 0x8000001A  STATUS\_NO\_MORE\_ENTRIES | No account was found with the specified [**privilege**](#gt_d8092e10-b227-4b44-b015-511bb8178940). |

Processing:

This message takes three arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to POLICY\_VIEW\_LOCAL\_INFORMATION.[<63>](#Appendix_A_63" \o "Product behavior note 63)

*UserRight*: A string representation of an account right. If the server does not recognize the account right, it MUST return STATUS\_NO\_SUCH\_PRIVILEGE.

The server executes the request by going through all accounts in its policy database and returning a set of all account object [**SIDs**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) that have that right or privilege.

*EnumerationBuffer*: Used to return a set of account SIDs that have the specified UserRight.

##### LsarEnumerateAccountRights (Opnum 36)

The LsarEnumerateAccountRights method is invoked to retrieve a list of rights associated with an existing account.

1. NTSTATUS LsarEnumerateAccountRights(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_SID AccountSid,
4. [out] PLSAPR\_USER\_RIGHT\_SET UserRights
5. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**AccountSid:** A [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b) that the caller is inquiring about.

**UserRights:** Used to return a list of right names associated with the account.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One or more of the supplied parameters was invalid. |
| 0xC0000034  STATUS\_OBJECT\_NAME\_NOT\_FOUND | The specified account object does not exist. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message takes two arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to ACCOUNT\_VIEW.

*AccountSid*: A SID of the account to query. The server MUST verify that the SID pointed to by *AccountSid* is valid and fail the request with STATUS\_INVALID\_PARAMETER otherwise. If IsRequestorAnonymous() returns TRUE (section [3.1.4.2.3](#Section_5d50b55fe9c74af6bf8502e8043f66ea)) and LsaRestrictAnonymous is set to TRUE, the call MUST fail with STATUS\_OBJECT\_NAME\_NOT\_FOUND. The server MUST verify that such an account exists in its database and fail the request with STATUS\_OBJECT\_NAME\_NOT\_FOUND otherwise.

The server MUST return the string names of all the system access rights and [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) associated with the account. It is valid for the server to return an empty set if the account object does not contain any rights.

##### LsarAddAccountRights (Opnum 37)

The LsarAddAccountRights method is invoked to add new rights to an [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b). If the account object does not exist, the system will attempt to create one.

1. NTSTATUS LsarAddAccountRights(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_SID AccountSid,
4. [in] PLSAPR\_USER\_RIGHT\_SET UserRights
5. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**AccountSid:** A [**security identifier**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of an account to add the rights to.

**UserRights:** A set of right names to add to the account.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One or more of the supplied parameters was invalid. |
| 0xC0000060  STATUS\_NO\_SUCH\_PRIVILEGE | The rights supplied were not recognized. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message takes three arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with:

* RequiredAccess set to POLICY\_CREATE\_ACCOUNT if the account identified by the *AccountSid* parameter does not exist in the server's database, or
* RequiredAccess set to ACCOUNT\_ADJUST\_PRIVILEGES | ACCOUNT\_ADJUST\_SYSTEM\_ACCESS | ACCOUNT\_VIEW if the account identified by the *AccountSid* parameter exists in the server's database

*AccountSid*: A security identifier of the account object. The server MUST create the account object if one does not exist.

*UserRights*: A set of system access rights and [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) to be added to the account. If the server does not recognize any of the rights, it MUST return STATUS\_NO\_SUCH\_PRIVILEGE.

##### LsarRemoveAccountRights (Opnum 38)

The LsarRemoveAccountRights method is invoked to remove rights from an [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b).

1. NTSTATUS LsarRemoveAccountRights(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_SID AccountSid,
4. [in] unsigned char AllRights,
5. [in] PLSAPR\_USER\_RIGHT\_SET UserRights
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**AccountSid:** A [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) of an account object.

**AllRights:** If this field is not set to 0, all rights will be removed.

**UserRights:** A set of rights to remove from the account.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One or more of the supplied parameters was invalid. |
| 0xC0000060  STATUS\_NO\_SUCH\_PRIVILEGE | The rights supplied were not recognized. |
| 0xC0000034  STATUS\_OBJECT\_NAME\_NOT\_FOUND | An account with this [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) does not exist. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC00000BB  STATUS\_NOT\_SUPPORTED | The operation is not supported by the server. |

Processing:

This message takes four arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to ACCOUNT\_ADJUST\_PRIVILEGES | ACCOUNT\_ADJUST\_SYSTEM\_ACCESS | ACCOUNT\_VIEW | DELETE.

If IsRequestorAnonymous() returns TRUE (section [3.1.4.2.3](#Section_5d50b55fe9c74af6bf8502e8043f66ea)) and LsaRestrictAnonymous is set to TRUE, the call MUST fail with STATUS\_OBJECT\_NAME\_NOT\_FOUND.

*AccountSid*: The security identifier of the account to modify. The server MUST verify that such an account exists in its database and fail the request with STATUS\_OBJECT\_NAME\_NOT\_FOUND otherwise.

*AllRights*: If nonzero, all system access rights and [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) will be stripped from the account.

*UserRights*: A set of rights and privileges to remove from the account. If the server does not recognize any of the rights, server MUST return STATUS\_NO\_SUCH\_PRIVILEGE.

The server MUST NOT allow removal of "SeAuditPrivilege", "SeChangeNotifyPrivilege", "SeImpersonatePrivilege", and "SeCreateGlobalPrivilege" from accounts represented with SIDs "S-1-5-19" and "S-1-5-20". The request MUST be rejected with STATUS\_NOT\_SUPPORTED.[<64>](#Appendix_A_64" \o "Product behavior note 64)

If the resulting set of access rights and privileges is empty, the server MUST delete the account object from its database.

#### Secret Object Methods

The message processing of methods in this section MUST use the abstract data model defined in section [3.1.1.4](#Section_483f1b6e7b1443419ab29b99c01f896e).

| Method (opnum) | Summary |
| --- | --- |
| LsarCreateSecret (opnum 16) | Creates a new [**secret object**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d) in the policy database. |
| LsarOpenSecret (opnum 28) | Opens a handle to an existing secret object. |
| LsarSetSecret (opnum 29) | Sets the value of the secret object. |
| LsarQuerySecret (opnum 30) | Retrieves the value of the secret object. |
| LsarStorePrivateData (opnum 42) | Stores private data in the server's policy database as a secret object. |
| LsarRetrievePrivateData (opnum 43) | Retrieves private data from a secret object in the server's policy database. |

The server SHOULD[<65>](#Appendix_A_65" \o "Product behavior note 65) support the following methods:

* [LsarSetSecret](#Section_21c1a153032c4869afc9186b2346dfab)
* [LsarQuerySecret](#Section_e36cfffafd53437ea5a71a95cfdda4c1)
* [LsarStorePrivateData](#Section_b79c94fed7174ecf963ca200682921dc)
* [LsarRetrievePrivateData](#Section_b46f3725d3de46b78245a14edeb278a1)

If the server does not support these methods, the server MUST respond with an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) exception. If the server supports these methods, the server MUST perform the operations in the message processing section for each method.

##### LsarCreateSecret (Opnum 16)

The LsarCreateSecret method is invoked to create a new [**secret object**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d) in the server's database.

1. NTSTATUS LsarCreateSecret(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_UNICODE\_STRING SecretName,
4. [in] ACCESS\_MASK DesiredAccess,
5. [out] LSAPR\_HANDLE\* SecretHandle
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**SecretName:** The name of the secret object to be created.

**DesiredAccess:** A bitmask that specifies accesses to be granted to the newly created and opened secret object at this time.

**SecretHandle:** Used to return a handle to the newly created secret object.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied parameters is invalid. This can happen, for example, if *SecretHandle* is NULL or if *SecretName* is not a valid name for a secret object. Secret naming rules are specified in the processing rules shown below for the *SecretName* parameter. |
| 0xC0000035  STATUS\_OBJECT\_NAME\_COLLISION | The secret object by the specified name already exists. |
| 0xC0000106  STATUS\_NAME\_TOO\_LONG | The length of specified secret name exceeds the maximum set by the server. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message takes four arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to POLICY\_CREATE\_SECRET.

*SecretName*: Name of the secret object to be created. The server MUST verify that the string satisfies the RPC\_UNICODE\_STRING syntax restrictions specified in section [3.1.4.10](#Section_ba5f1e31b2a742bd9a903650a6e5f6f5), and fail the request with STATUS\_INVALID\_PARAMETER otherwise. The server MUST also check that the following constraints are satisfied by *SecretName*, and fail the request with STATUS\_INVALID\_PARAMETER if the name does not check out:

* Must not be empty.
* Must not contain the "\" character.[<66>](#Appendix_A_66" \o "Product behavior note 66)[<67>](#Appendix_A_67" \o "Product behavior note 67)[<68>](#Appendix_A_68" \o "Product behavior note 68)

*DesiredAccess*: Contains the access bits that the caller is asking to receive for the handle returned in *SecretHandle*. *DesiredAccess* is access-checked according to section [3.1.4.2.1](#Section_e5e1e32e4066435db669044fe997eaf7). The method-specific portion of the check is the following.

1. IF (IsRequestorAnonymous() and LsaRestrictAnonymous is set to TRUE) THEN
2. Return STATUS\_OBJECT\_NAME\_NOT\_FOUND
3. END IF

The valid secret-rights bits are specified in section [2.2.1.1.4](#Section_88C6BD186C404A82AE19FE7BFEC5108B), and the [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) is specified in section [3.1.1.4](#Section_483F1B6E7B1443419AB29B99C01F896E). The IsRequestorAnonymous procedure is specified in section [3.1.4.2.3](#Section_5d50b55fe9c74af6bf8502e8043f66ea).

*SecretHandle*: If the request is successful, this parameter is used to return a handle (section [3.1.1.7](#Section_1011130b0413460d81edd1821d141639)) to the newly created secret object with its fields initialized as follows:

* LsaContextHandle.HandleType = "Secret"
* LsaContextHandle.Object = the secret object
* LsaContextHandle.GrantedAccess = as specified in section 3.1.4.2.1

Both "current time" and "old time" attributes of a secret will be set to the server's current time at the instance of creation. Both "old value" and "current value" will be set to NULL until they are modified by the [LsarSetSecret](#Section_21c1a153032c4869afc9186b2346dfab) message.

The server MUST check that the secret by the name *SecretName* does not already exist and fail the request with STATUS\_OBJECT\_NAME\_COLLISION otherwise.[<69>](#Appendix_A_69" \o "Product behavior note 69)

##### LsarOpenSecret (Opnum 28)

The LsarOpenSecret method is invoked to obtain a handle to an existing [**secret object**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d).

1. NTSTATUS LsarOpenSecret(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_UNICODE\_STRING SecretName,
4. [in] ACCESS\_MASK DesiredAccess,
5. [out] LSAPR\_HANDLE\* SecretHandle
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**SecretName:** The name of the secret object to open.

**DesiredAccess:** The requested type of access.

**SecretHandle:** Used to return the handle to the opened secret object.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC0000034  STATUS\_OBJECT\_NAME\_NOT\_FOUND | The secret with the specified name was not found. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | Some of the parameters supplied were invalid. |

Processing:

This message takes four arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. *PolicyHandle*.GrantedAccess MUST NOT be considered for this call because the access check MUST happen on the secret object.

*SecretName*: The name of the secret to be opened. The server MUST verify that the name syntax restrictions on secrets specified in section [3.1.4.6.1](#Section_35a984a1d0024d60946db557ff4c46e0) are satisfied, and fail the request with STATUS\_INVALID\_PARAMETER otherwise. The server MUST verify that the secret object with this name exists in its policy database and fail the request with STATUS\_OBJECT\_NAME\_NOT\_FOUND otherwise.[<70>](#Appendix_A_70" \o "Product behavior note 70)

*DesiredAccess*: A bitmask specifying the type of access that the caller attempts to obtain from the secret object, which is access-checked according to section [3.1.4.2.1](#Section_e5e1e32e4066435db669044fe997eaf7). The method-specific portion of the check is as follows:

1. IF (IsRequestorAnonymous() and LsaRestrictAnonymous is set to TRUE) THEN
2. Return STATUS\_OBJECT\_NAME\_NOT\_FOUND
3. END IF

The valid secret-rights bits are specified in section [2.2.1.1.4](#Section_88C6BD186C404A82AE19FE7BFEC5108B) and the [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) is specified in section [3.1.1.4](#Section_483F1B6E7B1443419AB29B99C01F896E). The IsRequestorAnonymous procedure is specified in section [3.1.4.2.3](#Section_5d50b55fe9c74af6bf8502e8043f66ea).

*SecretHandle*: If the request is successful, this parameter is used to return a handle (section [3.1.1.7](#Section_1011130b0413460d81edd1821d141639)) to the opened secret object with its fields initialized as follows:

* LsaContextHandle.HandleType = "Secret"
* LsaContextHandle.Object = the secret object
* LsaContextHandle.GrantedAccess = as specified in section 3.1.4.2.1

##### LsarSetSecret (Opnum 29)

The LsarSetSecret method is invoked to set the current and old values of the [**secret object**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d).

1. NTSTATUS LsarSetSecret(
2. [in] LSAPR\_HANDLE SecretHandle,
3. [in, unique] PLSAPR\_CR\_CIPHER\_VALUE EncryptedCurrentValue,
4. [in, unique] PLSAPR\_CR\_CIPHER\_VALUE EncryptedOldValue
5. );

**SecretHandle:** An open secret object handle.

**EncryptedCurrentValue:** A binary large object (BLOB) representing a new encrypted cipher value. It is valid for this parameter to be NULL, in which case the value is deleted from the server's policy database.

**EncryptedOldValue:** A BLOB representing the encrypted old value. It is valid for this parameter to be NULL, in which case the current value in the policy database is copied.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One or more of the supplied parameters was invalid. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *SecretHandle* is not a valid handle. |

Processing:

This message contains three input parameters:

*SecretHandle*: An open handle to a secret object. If the handle is not a valid context handle to a secret object or *SecretHandle*.HandleType does not equal "Secret", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *SecretHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to SECRET\_SET\_VALUE.[<71>](#Appendix_A_71" \o "Product behavior note 71)

*EncryptedCurrentValue*: The version of the new secret value that is being set, encrypted as specified in section [5.1.2](#Section_cce8ae7801384129954ec65e0c0bffed). It is valid for this parameter to be NULL, in which case the server MUST delete the current value in its database. If decryption fails, the server must return an implementation-specific error.[<72>](#Appendix_A_72" \o "Product behavior note 72)

*EncryptedOldValue*: The version of the old secret value that is being set, encrypted as specified in section 5.1.2. It is valid for this parameter to be NULL, in which case the server MUST delete the old value in its database and replace it with the previous version of "CurrentValue". If decryption fails, the server must return an implementation-specific error.[<73>](#Appendix_A_73" \o "Product behavior note 73)

The server MUST also maintain "time stamp" values for current and old values of the secret object. The following table lists the rules by which the time stamps are computed.

| Old secret value | New secret value | Effect on old time | Effect on new time |
| --- | --- | --- | --- |
| NULL | NULL | Old value of "new secret time" | Current server time |
| NULL | Non-NULL | Old value of "new secret time" | Current server time |
| Non-NULL | NULL | Current server time | Current server time |
| Non-NULL | Non-NULL | Current server time | Current server time |

##### LsarQuerySecret (Opnum 30)

The LsarQuerySecret method is invoked to retrieve the current and old (or previous) value of the [**secret object**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d).

1. NTSTATUS LsarQuerySecret(
2. [in] LSAPR\_HANDLE SecretHandle,
3. [in, out, unique] PLSAPR\_CR\_CIPHER\_VALUE\* EncryptedCurrentValue,
4. [in, out, unique] PLARGE\_INTEGER CurrentValueSetTime,
5. [in, out, unique] PLSAPR\_CR\_CIPHER\_VALUE\* EncryptedOldValue,
6. [in, out, unique] PLARGE\_INTEGER OldValueSetTime
7. );

**SecretHandle:** An open secret object handle.

**EncryptedCurrentValue:** Used to return the encrypted current value of the secret object.

**CurrentValueSetTime:** Used to return the time when the current value was set.

**EncryptedOldValue:** A BLOB representing the encrypted old value. It is valid for this parameter to be NULL, in which case the current value in the policy database is copied.

**OldValueSetTime:** The time corresponding to the instant that the old value was last changed.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *SecretHandle* is not a valid handle. |

Processing:

This message takes five arguments:

*SecretHandle*: An open handle to a secret object. If the handle is not a valid context handle to a secret object or *SecretHandle*.HandleType does not equal "Secret", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *SecretHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to SECRET\_QUERY\_VALUE.[<74>](#Appendix_A_74" \o "Product behavior note 74)

*EncryptedCurrentValue*: Used to return the current value of the secret, encrypted as specified in section [5.1.2](#Section_cce8ae7801384129954ec65e0c0bffed). This parameter can be NULL if the caller is not interested in this information.[<75>](#Appendix_A_75" \o "Product behavior note 75)

*CurrentValueSetTime:*  The time corresponding to the instant that the current value was last changed. This parameter can be NULL if the caller is not interested in this information.

*EncryptedOldValue*: Used to return the old value of the secret, encrypted as specified in section 5.1.2. This parameter can be NULL if the caller is not interested in this information.[<76>](#Appendix_A_76" \o "Product behavior note 76)

*OldValueSetTime*: The time corresponding to the instance that the old value was last changed. This parameter can be NULL if the caller is not interested in this information.

##### LsarStorePrivateData (Opnum 42)

The LsarStorePrivateData method is invoked to store a secret value.

1. NTSTATUS LsarStorePrivateData(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_UNICODE\_STRING KeyName,
4. [in, unique] PLSAPR\_CR\_CIPHER\_VALUE EncryptedData
5. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**KeyName:** The name under which private data will be stored.

**EncryptedData:** The secret value to be stored.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One or more of the supplied parameters was invalid. |

Processing:

This message takes three arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to POLICY\_CREATE\_SECRET.

*KeyName*: A string identifying the name of the [**secret object**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d) under which the private data would be stored. The server MUST verify that *KeyName* is syntactically valid and reject the request with STATUS\_INVALID\_PARAMETER otherwise. If a secret object by this name does not exist and the *EncryptedData* parameter is not NULL, the server MUST verify that the caller has POLICY\_CREATE\_SECRET access. If the secret does exist and the *EncryptedData* parameter is not NULL, the access check is performed for the SECRET\_SET\_VALUE right against the secret's [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350). If the access check fails, the server MUST return STATUS\_ACCESS\_DENIED. If the *EncryptedData* parameter is NULL, the server MUST check that the caller has DELETE access to the secret object and, if so, delete the secret object from the policy database.

*EncryptedData*: The value of the secret to be stored. This value is encrypted as specified in section [5.1.2](#Section_cce8ae7801384129954ec65e0c0bffed). As mentioned already, a caller that wants the secret to be deleted simply passes NULL for this value. If decryption fails, the server must return an implementation-specific error.[<77>](#Appendix_A_77" \o "Product behavior note 77)

##### LsarRetrievePrivateData (Opnum 43)

The LsarRetrievePrivateData method is invoked to retrieve a secret value.

1. NTSTATUS LsarRetrievePrivateData(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_UNICODE\_STRING KeyName,
4. [in, out] PLSAPR\_CR\_CIPHER\_VALUE\* EncryptedData
5. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**KeyName:** The name identifying the secret value to be retrieved.

**EncryptedData:** Receives the encrypted value of the [**secret object**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d).

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied parameters was invalid. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC0000034  STATUS\_OBJECT\_NAME\_NOT\_FOUND | The key with the specified name was not found. |

Processing:

This message takes three arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to SECRET\_QUERY\_VALUE.

*KeyName*: A string identifying the name of the secret object to be queried. If IsRequestorAnonymous() returns TRUE (section [3.1.4.2.3](#Section_5d50b55fe9c74af6bf8502e8043f66ea)) and LsaRestrictAnonymous is set to TRUE, the call MUST fail with STATUS\_OBJECT\_NAME\_NOT\_FOUND. If a secret object by this name does not exist, the server MUST return STATUS\_OBJECT\_NAME\_NOT\_FOUND.

*EncryptedData*: Used to return an encrypted version of the secret value. This value is encrypted as specified in section [5.1.2](#Section_cce8ae7801384129954ec65e0c0bffed).

#### Trusted Domain Object Methods

Trusted domain objects SHOULD be created only on a server implementation that is in the [**domain controller**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) configuration.[<78>](#Appendix_A_78" \o "Product behavior note 78)

The message processing of methods in this section MUST use the abstract data model as specified in section [3.1.1.5](#Section_0228f75e9725479cb4cd1cacd667343c).

| Method (opnum) | Summary |
| --- | --- |
| LsarCreateTrustedDomainEx2 (opnum 59) | Creates a new [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4) in the server's policy database. |
| LsarCreateTrustedDomainEx (opnum 51) | Superseded by [LsarCreateTrustedDomainEx2](#Section_cc86a55db61948fd998a65cca15efeb9). |
| LsarCreateTrustedDomain (opnum 12) | Superseded by LsarCreateTrustedDomainEx2. |
| LsarOpenTrustedDomain (opnum 25) | Opens a handle to an existing trusted domain object that matches the given [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) [**security identifier**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d). |
| LsarOpenTrustedDomainByName (opnum 55) | Opens a handle to an existing trusted domain object that matches the given DNS or NetBIOS name. |
| LsarQueryTrustedDomainInfo (opnum 39) | Obtains information about a trusted domain object. |
| LsarSetTrustedDomainInfo (opnum 40) | Sets information on a trusted domain object. |
| LsarSetTrustedDomainInfoByName (opnum 49) | Sets information on a trusted domain object without having to first open a handle to it. |
| LsarSetInformationTrustedDomain (opnum 27) | Sets information on a trusted domain object. |
| LsarQueryTrustedDomainInfoByName (opnum 48) | Obtains information about a trusted domain object without having to first open a handle to it. |
| LsarQueryInfoTrustedDomain (opnum 26) | Obtains information about a trusted domain object. |
| LsarDeleteTrustedDomain (opnum 41) | Removes a trusted domain object from the server's policy database. |
| LsarEnumerateTrustedDomainsEx (opnum 50) | Enumerates all trusted domain objects in the server's policy database. |
| LsarEnumerateTrustedDomains (opnum 13) | Enumerates trusted domain objects in the server's policy database. |
| LsarQueryForestTrustInformation (opnum 73) | Obtains information from a trusted domain object corresponding to a [**forest trust**](#gt_035d9ce5-f117-4251-8d4d-127c462ec4a0) relationship. |
| LsarSetForestTrustInformation (opnum 74) | Sets information on a trusted domain object corresponding to a cross-forest trust relationship. |

##### LsarOpenTrustedDomain (Opnum 25)

The LsarOpenTrustedDomain method is invoked to obtain a handle to a [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4).

1. NTSTATUS LsarOpenTrustedDomain(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_SID TrustedDomainSid,
4. [in] ACCESS\_MASK DesiredAccess,
5. [out] LSAPR\_HANDLE\* TrustedDomainHandle
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**TrustedDomainSid:** A [**security identifier**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4) that is being opened.

**DesiredAccess:** A bitmask of access rights to open the object with.

**TrustedDomainHandle:** Used to return the trusted domain object handle.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied parameters is invalid. For instance, this can happen if the security identifier *TrustedDomainSid* is not a valid [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) security identifier. Section [3.1.4.10](#Section_BA5F1E31B2A742BD9A903650A6E5F6F5) specifies data validation rules, including what constitutes a valid domain security identifier. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC00000DF  STATUS\_NO\_SUCH\_DOMAIN | The specified trusted domain object does not exist. |
| 0xC00002B1  STATUS\_DIRECTORY\_SERVICE\_REQUIRED | The [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) was not available on the server. |

Processing:

If Active Directory is not running on this machine, the server MUST return STATUS\_DIRECTORY\_SERVICE\_REQUIRED.

This message takes four arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. *PolicyHandle*.GrantedAccess MUST NOT be considered for this call because the access check MUST happen on the trusted domain object.

*TrustedDomainSid*: A SID of the trusted domain object. The server MUST verify that the SID is a valid domain SID and reject the request with STATUS\_INVALID\_PARAMETER otherwise. If the trusted domain object with this SID does not exist, the server MUST fail the request with STATUS\_NO\_SUCH\_DOMAIN error code.

*DesiredAccess*: A bitmask specifying the type of access the caller attempts to obtain from the trusted domain object, which is access-checked according to section [3.1.4.2.1](#Section_e5e1e32e4066435db669044fe997eaf7). There is no method-specific portion of the check. The valid trusted-domain-rights bits are specified in section [2.2.1.1.5](#Section_E035F552031348B79BCAFDD9FD4E948E), and the [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) is specified in section [3.1.1.5](#Section_0228F75E9725479CB4CD1CACD667343C).

*TrustedDomainHandle*: If the request is successful, this parameter is used to return a handle (section [3.1.1.7](#Section_1011130b0413460d81edd1821d141639)) to the opened trusted domain object with its fields initialized as follows:[<79>](#Appendix_A_79" \o "Product behavior note 79)[<80>](#Appendix_A_80" \o "Product behavior note 80)

* LsaContextHandle.HandleType = "Trusted Domain"
* LsaContextHandle.Object = the trusted domain object
* LsaContextHandle.GrantedAccess = as specified in section 3.1.4.2.1

##### LsarQueryTrustedDomainInfo (Opnum 39)

The LsarQueryTrustedDomainInfo method is invoked to retrieve information on a [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4).

1. NTSTATUS LsarQueryTrustedDomainInfo(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_SID TrustedDomainSid,
4. [in] TRUSTED\_INFORMATION\_CLASS InformationClass,
5. [out, switch\_is(InformationClass)]
6. PLSAPR\_TRUSTED\_DOMAIN\_INFO\* TrustedDomainInformation
7. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**TrustedDomainSid:** A [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) of the trusted domain object.

**InformationClass:** Identifies the type of information the caller is interested in.

**TrustedDomainInformation:** Used to return the information on the trusted domain object to the caller.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One or more of the supplied parameters was invalid. |
| 0xC0000002  STATUS\_NOT\_IMPLEMENTED | The specified information class is not supported. |
| 0xC0000003  STATUS\_INVALID\_INFO\_CLASS | The *InformationClass* argument is outside the allowed range. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC00000DF  STATUS\_NO\_SUCH\_DOMAIN | The specified trusted domain object does not exist. |
| 0xC00002B1  STATUS\_DIRECTORY\_SERVICE\_REQUIRED | The [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) was not available on the server. |

Processing:

If Active Directory is not running on this machine, the server MUST return STATUS\_DIRECTORY\_SERVICE\_REQUIRED.

This message takes four arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set as specified in section [3.1.4.7.13](#Section_e74460c7db0345c3ac3ca72a840e4943).

*TrustedDomainSid*: The [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the trusted domain object to query. The server MUST verify that the caller has supplied a valid [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) SID for this parameter and fail the request with STATUS\_INVALID\_PARAMETER if the check fails. The server MUST verify that a trusted domain object with this SID exists in its policy database and fail the request with STATUS\_NO\_SUCH\_DOMAIN otherwise.

*InformationClass*: A value from the [TRUSTED\_INFORMATION\_CLASS](#Section_360691136c3845e8920e17f8ef36f578) enumeration that specifies which type of information the caller is requesting. Not all values are valid. For values outside the TRUSTED\_INFORMATION\_CLASS enumeration range, the server MUST reject the request with STATUS\_INVALID\_PARAMETER. For *InformationClass* values TrustedControllersInformation, TrustedDomainAuthInformationInternal, TrustedDomainFullInformationInternal, and for any values that would be rejected by an LsarQueryInfoTrustedDomain call, the server MUST reject the request with an implementation-specific error. For all other *InformationClass* values, the server MUST behave as if it is processing an LsarQueryInfoTrustedDomain call with a [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4) handle to the trusted domain identified by the *TrustedDomainSid* parameter.

*TrustedDomainInformation*: Used to return the requested information.

##### LsarSetTrustedDomainInfo (Opnum 40)

The LsarSetTrustedDomainInfo method is invoked to set information on a [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4). In some cases, if the trusted domain object does not exist, it will be created.

1. NTSTATUS LsarSetTrustedDomainInfo(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_SID TrustedDomainSid,
4. [in] TRUSTED\_INFORMATION\_CLASS InformationClass,
5. [in, switch\_is(InformationClass)]
6. PLSAPR\_TRUSTED\_DOMAIN\_INFO TrustedDomainInformation
7. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**TrustedDomainSid:** A [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the trusted domain object to be modified.

**InformationClass:** Identifies the type of information to be set on the trusted domain object.

**TrustedDomainInformation:** Information to be set on the trusted domain object.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One or more of the supplied parameters was invalid. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC00000DF  STATUS\_NO\_SUCH\_DOMAIN | The specified trusted domain object does not exist. |
| 0xC00002B1  STATUS\_DIRECTORY\_SERVICE\_REQUIRED | The [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) was not available on the server. |

Processing:

If Active Directory is not running on this machine, the server MUST return STATUS\_DIRECTORY\_SERVICE\_REQUIRED.

This method is similar to the [LsarSetInformationTrustedDomain](#Section_9ea46cefcc724109ba1391eda6b713bc) method, with some important differences. For one, this method takes a policy object handle instead of a trusted domain object handle. Another important distinction is that for some information classes this method, unlike LsarSetInformationTrustedDomain, will create a trusted domain object if one does not exist already.

This message takes four arguments:

*PolicyHandle*: An open handle to the policy object. The access rights required to perform the operation depend on the value of the *InformationClass* parameter. The access bits required for each information class are specified in section 3.1.4.7.14. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. *PolicyHandle*.GrantedAccess MUST NOT be considered for this call because the access check MUST happen on the trusted domain object. If the server is a [**read-only domain controller**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870), it MUST return an error.[<81>](#Appendix_A_81" \o "Product behavior note 81)

*TrustedDomainSid*: A SID of the trusted domain object to modify. The server MUST verify that the caller has supplied a valid [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) SID for this parameter and fail the request with STATUS\_INVALID\_PARAMETER if the check fails.

*InformationClass*: A value from the [TRUSTED\_INFORMATION\_CLASS](#Section_360691136c3845e8920e17f8ef36f578) enumeration that specifies which type of information the caller is setting. Not all *InformationClass* values are valid. The valid *InformationClass* values for this method are as follows:

* TrustedDomainNameInformation: The server MUST act as if an [LsarCreateTrustedDomain](#Section_373a4b1e1e8d45729c250bd7b045d3a3) message came in with its *TrustedDomainInformation.Name* parameter as the name passed in the *TrustedDomainInformation* parameter, its *TrustedDomainInformation.Sid* parameter as the SID passed in the *TrustedDomainSid* parameter, and its *DesiredAccess* parameter set to zero.
* TrustedPosixOffsetInformation: The server MUST verify that a trusted domain object with this SID exists in its policy database. If the object does not exist, the call MUST fail with STATUS\_NO\_SUCH\_DOMAIN. Otherwise, the server MUST verify that the caller has access to the trusted domain object as specified in section [3.1.4.2.1](#Section_e5e1e32e4066435db669044fe997eaf7) with *DesiredAccess* set to TRUSTED\_SET\_POSIX. There is no method-specific portion of this check.

Then the server MUST act as if an LsarSetInformationTrustedDomain message is being processed.

The server MAY support the following *InformationClass* values.[<82>](#Appendix_A_82" \o "Product behavior note 82) If the server does not support these values, it MUST return STATUS\_INVALID\_PARAMETER. If the server supports these values, it MUST perform the corresponding operations:

* TrustedDomainInformationEx: The server MUST check that a trusted domain object with this SID exists in its policy database. If the object does not exist, the server MUST create a new trusted domain object using the same processing rules as [LsarCreateTrustedDomainEx2](#Section_cc86a55db61948fd998a65cca15efeb9), and using the following parameters for the LsarCreateTrustedDomainEx2 processing rules:
  + *PolicyHandle* set to the same *PolicyHandle* in the original message.
  + *TrustedDomainInformation* set to the same *TrustedDomainInformation* in the original message.
  + *AuthenticationInformation* set to NULL.
  + *DesiredAccess* set to zero.

If the object does exist, the server MUST set the [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4) information using the same processing rules as LsarSetInformationTrustedDomain, and using the following parameters for the LsarSetInformationTrustedDomain processing rules:

* + *TrustDomainHandle* set to the handle to the trusted domain object.
  + *InformationClass* set to the same *InformationClass* in the original message.
  + *TrustedDomainInformation* set to the same *TrustedDomainInformation* in the original message.
* TrustedPasswordInformation: The server MUST verify that a trusted domain object with this SID exists in its policy database. If the object does not exist, the call MUST fail with STATUS\_NO\_SUCH\_DOMAIN. Otherwise, the server MUST open the [**secret object**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d), as defined in section [3.1.1.4](#Section_483f1b6e7b1443419ab29b99c01f896e), (or create a secret object, if one does not already exist) with "Name" set to "G$$<Trusted Domain Name>". The server MUST then set "Old Value" of the secret object to the "OldPassword" value in *TrustedDomainInformation* and set "New Value" of the secret object to the "Password" value in *TrustedDomainInformation*, similar to the processing when an [LsarSetSecret](#Section_21c1a153032c4869afc9186b2346dfab) request has been made.

The server MUST return STATUS\_INVALID\_PARAMETER for all other *InformationClass* arguments.

*TrustedDomainInformation*: Contains the data supplied by the caller to be set on the trusted domain object.

##### LsarDeleteTrustedDomain (Opnum 41)

The LsarDeleteTrustedDomain method is invoked to delete a [**trusted domain object (TDO)**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4).

1. NTSTATUS LsarDeleteTrustedDomain(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_SID TrustedDomainSid
4. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**TrustedDomainSid:** A [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) of the TDO to be deleted.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC00000DF  STATUS\_NO\_SUCH\_DOMAIN | The specified TDO does not exist. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One or more of the supplied parameters was invalid. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC00002B1  STATUS\_DIRECTORY\_SERVICE\_REQUIRED | The [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) was not available on the server. |
| 0xC0000403  STATUS\_USER\_DELETE\_TRUST\_QUOTA\_EXCEEDED | The caller's quota for the maximum allowed number of deleted TDOs is exceeded. |

Processing:

If Active Directory is not running on this machine, the server MUST return STATUS\_DIRECTORY\_SERVICE\_REQUIRED.

If the number of deleted TDOs that were created by the caller through the control access right Create-Inbound-Trust (defined in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 5.1.3.2.1) exceeds the value in the msDS-PerUserTrustTombstonesQuota attribute of the [**domain naming context (domain NC)**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) root object (defined in [MS-ADTS] section 6.1.1.1.4), the server MUST return STATUS\_USER\_DELETE\_TRUST\_QUOTA\_EXCEEDED. For the syntax of the msDS-PerUserTrustTombstonesQuota attribute, refer to [[MS-ADA2]](%5bMS-ADA2%5d.pdf#Section_e20ebc4e528540bab3bdffcb81c2783e) section 2.411. The server MUST enforce the quota check only for the TDOs created by control access right Create-Inbound-Trust and if the caller is the creator of that TDO.

This message takes two arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to TRUSTED\_QUERY\_DOMAIN\_NAME | DELETE.

*TrustedDomainSid*: The [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of a TDO to be deleted. The server MUST verify that the caller has supplied a valid [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) SID for this parameter and fail the request with STATUS\_INVALID\_PARAMETER if the check fails. The server MUST verify that a TDO with this SID exists in its policy database and fail the request with STATUS\_NO\_SUCH\_DOMAIN otherwise.

If the server is a [**read-only domain controller**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870), it MUST return an error.[<83>](#Appendix_A_83" \o "Product behavior note 83)

The server MUST also check whether a secret with name "G$$<Trusted Domain Name>" exists or not. If it exists, the server MUST delete that secret along with the [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4).

The server MUST also check whether an [**interdomain trust account**](#gt_ba0d31d7-aa03-4e10-936a-a0ebd276ebc9) with name "<Trusted Domain NetBIOS Name>$" exists. If it exists, the server MUST delete that account along with the trusted domain.

##### LsarQueryTrustedDomainInfoByName (Opnum 48)

The LsarQueryTrustedDomainInfoByName method is invoked to retrieve information about a [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4) by its string name.

1. NTSTATUS LsarQueryTrustedDomainInfoByName(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_UNICODE\_STRING TrustedDomainName,
4. [in] TRUSTED\_INFORMATION\_CLASS InformationClass,
5. [out, switch\_is(InformationClass)]
6. PLSAPR\_TRUSTED\_DOMAIN\_INFO\* TrustedDomainInformation
7. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**TrustedDomainName:** The name of the trusted domain object to query.

**InformationClass:** One of the [TRUSTED\_INFORMATION\_CLASS](#Section_360691136c3845e8920e17f8ef36f578) values identifying the type of information the caller is interested in.

**TrustedDomainInformation:** Used to return the information requested by the caller.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied parameters was invalid. |
| 0xC0000034  STATUS\_OBJECT\_NAME\_NOT\_FOUND | The trusted domain object with the specified name could not be found. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message is identical in its operation to [LsarQueryInfoTrustedDomain](#Section_e74460c7db0345c3ac3ca72a840e4943); the only exception is that the *TrustedDomainName* parameter is used to locate the trusted domain object, rather than having the caller supply the trusted domain object handle.

The trusted domain object is located by matching the *TrustedDomainName* parameter against the trusted domain object in the server's policy database. The trailing period on [**DNS names**](#gt_102a36e2-f66f-49e2-bee3-558736b2ecd5) is ignored for the purposes of comparison.

##### LsarSetTrustedDomainInfoByName (Opnum 49)

The LsarSetTrustedDomainInfoByName method is invoked to set information about a [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4) by its string name.

1. NTSTATUS LsarSetTrustedDomainInfoByName(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_UNICODE\_STRING TrustedDomainName,
4. [in] TRUSTED\_INFORMATION\_CLASS InformationClass,
5. [in, switch\_is(InformationClass)]
6. PLSAPR\_TRUSTED\_DOMAIN\_INFO TrustedDomainInformation
7. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**TrustedDomainName:** The name of the trusted domain object to set information on.

**InformationClass:** One of the [TRUSTED\_INFORMATION\_CLASS](#Section_360691136c3845e8920e17f8ef36f578) values indicating the type of information the caller is trying to set.

**TrustedDomainInformation:** The data being set.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied arguments is invalid. |
| 0xC0000034  STATUS\_OBJECT\_NAME\_NOT\_FOUND | The trusted domain object with the specified name could not be found. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message is identical in its operation to [LsarSetInformationTrustedDomain](#Section_9ea46cefcc724109ba1391eda6b713bc); the only exception is that the *TrustedDomainName* parameter is used to locate the trusted domain object, rather than having the caller supply the trusted domain object handle.

The trusted domain object is located by matching the *TrustedDomainName* parameter against the trusted domain object in the server's policy database. The trailing period on [**DNS names**](#gt_102a36e2-f66f-49e2-bee3-558736b2ecd5) is ignored for the purposes of comparison.

##### LsarEnumerateTrustedDomainsEx (Opnum 50)

The LsarEnumerateTrustedDomainsEx method is invoked to enumerate [**trusted domain objects**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4) in the server's database. The method is designed to be invoked multiple times to retrieve the data in fragments.

1. NTSTATUS LsarEnumerateTrustedDomainsEx(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in, out] unsigned long\* EnumerationContext,
4. [out] PLSAPR\_TRUSTED\_ENUM\_BUFFER\_EX EnumerationBuffer,
5. [in] unsigned long PreferedMaximumLength
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**EnumerationContext:** Used to keep track of the state of the enumeration in cases where the caller obtains its information in several fragments.

**EnumerationBuffer:** Contains a fragment of requested information.

**PreferedMaximumLength:** A value that indicates the approximate size of the data to be returned.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0x8000001A  STATUS\_NO\_MORE\_ENTRIES | No more information is available. |
| 0x00000105  STATUS\_MORE\_ENTRIES | More information is available by calling this method again. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message takes four arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to POLICY\_VIEW\_LOCAL\_INFORMATION.

*EnumerationContext*: A number that indicates a starting index at which to begin the enumeration. The server MUST always return all trusted domain objects in the same order, starting at the object whose index is *EnumerationContext*. To initiate a new enumeration, the client sets *EnumerationContext* to zero; otherwise, the client sets *EnumerationContext* to a value returned by a previous call to the method.

The server MUST return STATUS\_INVALID\_PARAMETER if the *EnumerationContext* parameter is NULL.

*EnumerationBuffer*: Used to return the results of enumeration. The server MUST fill *EnumerationBuffer* with as many trusted domain objects as possible, as determined by *PreferedMaximumLength*. If the size of all remaining objects is less than or equal to *PreferedMaximumLength*, the server MUST fill *EnumerationBuffer* with all objects. If the size of all remaining objects is greater than *PreferedMaximumLength*, the server MUST fill *EnumerationBuffer* with objects such that the size of the trusted domain objects returned is greater than or equal to *PreferedMaximumLength*, but would be less than *PreferedMaximumLength* if the last object had not been added to *EnumerationBuffer*. If there are no more objects than are returned in *EnumerationBuffer*, the server MUST return STATUS\_NO\_MORE\_ENTRIES. If there are more database objects than are returned in *EnumerationBuffer*, the server MUST set the *EnumerationContext* value to the index value that would allow it to resume enumeration correctly when this method is called again, and the server MUST return STATUS\_MORE\_ENTRIES. Note that this return value is not an error status.

*PreferedMaximumLength*: An indication about the approximate size, in bytes, of the data to be returned. Any unsigned 32-bit value is valid for the *PreferedMaximumLength* parameter.

If [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) is not running on this machine, the server MUST fill 0 objects in *EnumerationBuffer*, and return STATUS\_NO\_MORE\_ENTRIES.

##### LsarEnumerateTrustedDomains (Opnum 13)

The LsarEnumerateTrustedDomains method is invoked to request a list of [**trusted domain objects**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4) in the server's database. The method can be called multiple times to return its output in fragments.

1. NTSTATUS LsarEnumerateTrustedDomains(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] [out] unsigned long \*EnumerationContext,
4. [out] PLSAPR\_TRUSTED\_ENUM\_BUFFER EnumerationBuffer,
5. [in] unsigned long PreferedMaximumLength
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**EnumerationContext:** A pointer to a context value that is used to resume enumeration, if necessary.

**EnumerationBuffer:** A pointer to a structure that will contain the results of the enumeration.

**PreferedMaximumLength:** A value that indicates the approximate size of the data to be returned.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC0000105  STATUS\_MORE\_ENTRIES | More information is available to successive calls. |
| 0xC000001A  STATUS\_NO\_MORE\_ENTRIES | No more entries are available from the enumeration. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message takes four arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to POLICY\_VIEW\_LOCAL\_INFORMATION.

*EnumerationContext*: A number that indicates a starting index at which to begin the enumeration. The server MUST always return all trusted domain objects in the same order, starting at the object whose index is *EnumerationContext*. To initiate a new enumeration, the client sets *EnumerationContext* to zero; otherwise, the client sets *EnumerationContext* to a value returned by a previous call to the method.

The server MUST return STATUS\_INVALID\_PARAMETER if the *EnumerationContext* parameter is NULL.

This method differs from the [LsarEnumerateTrustedDomainsEx](#Section_14e37cf7b090497ca2e297e8425532a2) method in one significant way-in mixed-mode [**forests**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62), this method returns to the caller an entire set of [**domains**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) within the forest by enumerating all the cross-referenced objects in [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) in addition to domains that are [**trusted**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) explicitly.

*EnumerationBuffer*: Used to return the results of enumeration. The server MUST fill *EnumerationBuffer* with as many trusted domain objects as possible, as determined by *PreferedMaximumLength*. If the size of all remaining objects is less than or equal to *PreferedMaximumLength*, the server MUST fill *EnumerationBuffer* with all objects. If the size of all remaining objects is greater than *PreferedMaximumLength*, the server MUST fill *EnumerationBuffer* with objects such that the size of the trusted domain objects returned is greater than or equal to *PreferedMaximumLength*, but would be less than *PreferedMaximumLength* if the last object had not been added to *EnumerationBuffer*. If there are no more objects than are returned in *EnumerationBuffer*, the server MUST return STATUS\_NO\_MORE\_ENTRIES. If there are more database objects than are returned in *EnumerationBuffer*, the server MUST set the *EnumerationContext* value to the index value that would allow it to resume enumeration correctly when this method is called again, and the server MUST return STATUS\_MORE\_ENTRIES. Note that this return value is not an error status.

When enumerating trusted domain objects for this message, the server MUST limit the trusted domain objects returned to the following subset only:

* Outbound Trusts: The trust direction has the TRUST\_DIRECTION\_OUTBOUND bit set.
* Uplevel or Downlevel Trusts: The trust type is TRUST\_TYPE\_DOWNLEVEL or TRUST\_TYPE\_UPLEVEL.
* Non-uplevel-only Trusts: The Trust Attributes field does not have the TRUST\_ATTRIBUTE\_UPLEVEL\_ONLY bit set.

Trust types and attributes are specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6.

*PreferedMaximumLength*: An indication about the approximate size, in bytes, of the data to be returned. Any unsigned 32-bit value is valid for the *PreferedMaximumLength* parameter.

If Active Directory is not running on this machine, the server MUST fill 0 objects in *EnumerationBuffer*, and return STATUS\_NO\_MORE\_ENTRIES.

##### LsarOpenTrustedDomainByName (Opnum 55)

The LsarOpenTrustedDomainByName method is invoked to open a [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4) handle by supplying the name of the [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4).

1. NTSTATUS LsarOpenTrustedDomainByName(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_UNICODE\_STRING TrustedDomainName,
4. [in] ACCESS\_MASK DesiredAccess,
5. [out] LSAPR\_HANDLE\* TrustedDomainHandle
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**TrustedDomainName:** The name of the trusted domain object.

**DesiredAccess:** The type of access requested by the caller.

**TrustedDomainHandle:** Used to return the opened trusted domain handle.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied arguments was invalid. |
| 0xC0000034  STATUS\_OBJECT\_NAME\_NOT\_FOUND | A trusted domain object by this name was not found. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message takes four arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. *PolicyHandle*.GrantedAccess MUST NOT be considered for this call because the access check MUST happen on the trusted domain object.

*TrustedDomainName*: Contains the name of the trusted domain to be opened. This can be a DNS or a NetBIOS name. If the server cannot locate a trusted domain object by this name in its policy database, the server MUST return STATUS\_OBJECT\_NAME\_NOT\_FOUND. The same status code MUST be returned by the server if [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) is not running on this machine.

*DesiredAccess*: The set of rights that the caller attempts to obtain from the trusted domain object, which is access-checked according to section [3.1.4.2.1](#Section_e5e1e32e4066435db669044fe997eaf7). There is no method-specific portion of the check. The valid trusted-domain-rights bits are specified in section [2.2.1.1.5](#Section_E035F552031348B79BCAFDD9FD4E948E), and the [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) is specified in section [3.1.1.5](#Section_0228F75E9725479CB4CD1CACD667343C).

*TrustedDomainHandle*: If the request is successful, this parameter is used to return a handle (section [3.1.1.7](#Section_1011130b0413460d81edd1821d141639)) to the opened trusted domain object with its fields initialized as follows:

* LsaContextHandle.HandleType = "Trusted Domain"
* LsaContextHandle.Object = the trusted domain object
* LsaContextHandle.GrantedAccess = as specified in section 3.1.4.2.1

##### LsarCreateTrustedDomainEx2 (Opnum 59)

The LsarCreateTrustedDomainEx2 method is invoked to create a new [**trusted domain object (TDO)**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4).[<84>](#Appendix_A_84" \o "Product behavior note 84)

1. NTSTATUS LsarCreateTrustedDomainEx2(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PLSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX TrustedDomainInformation,
4. [in] PLSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL AuthenticationInformation,
5. [in] ACCESS\_MASK DesiredAccess,
6. [out] LSAPR\_HANDLE\* TrustedDomainHandle
7. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**TrustedDomainInformation:** Information about the new TDO to be created.

**AuthenticationInformation:** Encrypted authentication information for the new TDO.

**DesiredAccess:** An access mask specifying desired access to the TDO handle.

**TrustedDomainHandle:** Used to return the handle for the newly created TDO.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied arguments is invalid. |
| 0xC0000300  STATUS\_NOT\_SUPPORTED\_ON\_SBS | The operation is not supported on a particular product.[<85>](#Appendix_A_85" \o "Product behavior note 85) |
| 0xC00000DD  STATUS\_INVALID\_DOMAIN\_STATE | The operation cannot complete in the current state of the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). |
| 0xC00002B1  STATUS\_DIRECTORY\_SERVICE\_REQUIRED | The [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) was not available on the server. |
| 0xC0000078  STATUS\_INVALID\_SID | The [**security identifier**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4) is not valid. |
| 0xC00002E9  STATUS\_CURRENT\_DOMAIN\_NOT\_ALLOWED | [**Trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) cannot be established with the current domain. |
| 0xC0000035  STATUS\_OBJECT\_NAME\_COLLISION | Another TDO already exists that matches some of the identifying information of the supplied information. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC0000401  STATUS\_PER\_USER\_TRUST\_QUOTA\_EXCEEDED | The caller's quota for maximum number of TDOs that can be created by control access right Create-Inbound-Trust is exceeded. |
| 0xC0000402  STATUS\_ALL\_USER\_TRUST\_QUOTA\_EXCEEDED | The combined users' quota for maximum number of TDOs that can be created by control access right Create-Inbound-Trust is exceeded. |

Processing:

If Active Directory is not running on this machine, the server MUST return STATUS\_DIRECTORY\_SERVICE\_REQUIRED.[<86>](#Appendix_A_86" \o "Product behavior note 86) If the server is a [**read-only domain controller**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870), it MUST return an error.[<87>](#Appendix_A_87" \o "Product behavior note 87)

If the caller is not a member of the Domain Admins group, the server MUST return STATUS\_ACCESS\_DENIED for policy handle access checking.

If the TDO creation failed due to the caller not having standard access rights to create the TDO, then the server MUST check the caller's control access right (defined in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 5.1.3.2.1). The TDO creation by control access right is allowed if:

* The trust is an inbound-only [**forest trust**](#gt_035d9ce5-f117-4251-8d4d-127c462ec4a0). The server MUST return STATUS\_ACCESS\_DENIED if the trust to be created is not an inbound-only forest trust.
* The caller has the control access right to create an inbound trust on the domain object.
* The caller's quota for trust object creations has not been exceeded. If the number of TDOs that have been created by the caller through control access right Create-Inbound-Trust exceeds the value in the msDS-PerUserTrustQuota attribute of the [**domain NC**](#gt_40a58fa4-953e-4cf3-96c8-57dba60237ef) root object, then the server MUST return STATUS\_PER\_USER\_TRUST\_QUOTA\_EXCEEDED. For the syntax of the msDS-PerUserTrustQuota attribute, refer to [[MS-ADA2]](%5bMS-ADA2%5d.pdf#Section_e20ebc4e528540bab3bdffcb81c2783e) section 2.410.
* The combined users' quota for trust object creations has not been exceeded. If the number of TDOs that have been created through control access right Create-Inbound-Trust exceeds the value in the msDS-AllUsersTrustQuota attribute of the domain NC root object (defined in [MS-ADTS] section 6.1.1.1.4), then the server MUST return STATUS\_ALL\_USER\_TRUST\_QUOTA\_EXCEEDED. For the syntax of the msDS-AllUsersTrustQuota attribute, refer to [MS-ADA2] section 2.212.

This message takes five arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. *PolicyHandle*.GrantedAccess MUST NOT be considered for this call because the access check MUST happen on the TDO.

*TrustedDomainInformation*: A structure containing most components of a TDO makeup. The data provided in this parameter MUST be checked for validity in accordance with rules for TDO consistency specified in "Trust Objects" in [MS-ADTS] section 6.1.6. The server MUST reject invalid input with STATUS\_INVALID\_PARAMETER. The server MUST return STATUS\_INVALID\_DOMAIN\_STATE in the following cases:

* The TrustAttributes TRUST\_ATTRIBUTE\_FOREST\_TRANSITIVE flag is set and the forestFunctionality specified in [MS-ADTS] section 3.1.1.3.2.27 is DS\_BEHAVIOR\_WIN2003 or higher.
* The TrustAttributes TRUST\_ATTRIBUTE\_FOREST\_TRANSITIVE flag is set and the **DnsForestName** and **DnsDomainName** fields in DNS Domain Information (see section [3.1.1.1](#Section_c939c70e8cf04d90a9cc1c5002951723)) do not contain the same value.
* The TrustAttributes TRUST\_ATTRIBUTE\_CROSS\_ORGANIZATION flag is set and the forestFunctionality specified in [MS-ADTS] section 3.1.1.3.2.27 is DS\_BEHAVIOR\_WIN2003 or higher.

If one or more properties in *TrustedDomainInformation* points to the current domain (such as the domain that the server is a part of), the server MUST return STATUS\_CURRENT\_DOMAIN\_NOT\_ALLOWED. If there is another domain that claims the same properties, the server MUST return STATUS\_OBJECT\_NAME\_COLLISION. Each field in this structure maps to a field in the TDO model, as specified in section [3.1.1.5](#Section_0228f75e9725479cb4cd1cacd667343c). If the operation succeeds, the server MUST update its database with a new TDO field populated from this input parameter.

*AuthenticationInformation*: A structure containing authentication information for the trusted domain. The server first MUST decrypt this data structure using an algorithm (as specified in section [5.1.1](#Section_1f5bd3edcfdd42aba2acf0786082bb21)) with the key being the session key negotiated by the transport. The server then MUST unmarshal the data inside this structure and then store it into a structure whose format is specified in section [2.2.7.11](#Section_084fdb6b5bc349129aed0257159996dd). This structure MUST then be stored on Trust Incoming and Outgoing Password properties.

*DesiredAccess*: A bitmask containing a set of access rights that the caller attempts to obtain from the TDO, which is access-checked as specified in section [3.1.4.2.1](#Section_e5e1e32e4066435db669044fe997eaf7). Whatever the set of access rights requested by the caller, the server MUST also set the TRUSTED\_SET\_AUTH bit inside *DesiredAccess* before performing the [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) check. There is no method-specific portion of the check.

The valid trusted-domain-rights bits are specified in section [2.2.1.1.5](#Section_E035F552031348B79BCAFDD9FD4E948E), and the security descriptor is specified in section 3.1.1.5.

*TrustedDomainHandle*: If the request is successful, this parameter is used to return a handle (section [3.1.1.7](#Section_1011130b0413460d81edd1821d141639)) to the newly created TDO with its fields initialized as follows:

* LsaContextHandle.HandleType = "Trusted Domain"
* LsaContextHandle.Object = the TDO
* LsaContextHandle.GrantedAccess = as specified in section 3.1.4.2.1

New TDOs are always created without [**forest trust information**](#gt_8c0b82d9-efec-4723-95a9-8564edf9ba44). The **ForestTrustInfo** and **ForestTrustLength** fields of the TDO are thus set to NULL and 0, respectively.

If the trust being created is inbound or bidirectional as defined in the **TrustDirection** field of the *TrustedDomainInformation* parameter, then the server MUST also update its database with a new [**interdomain trust account**](#gt_ba0d31d7-aa03-4e10-936a-a0ebd276ebc9) populated as specified in [MS-ADTS] section 6.1.6.8.

##### LsarCreateTrustedDomainEx (Opnum 51)

The LsarCreateTrustedDomainEx method is invoked to create a new [**trusted domain object (TDO)**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4).

1. NTSTATUS LsarCreateTrustedDomainEx(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PLSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX TrustedDomainInformation,
4. [in] PLSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION AuthenticationInformation,
5. [in] ACCESS\_MASK DesiredAccess,
6. [out] LSAPR\_HANDLE\* TrustedDomainHandle
7. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**TrustedDomainInformation:** Information about the new TDO to be created.

**AuthenticationInformation:** Encrypted authentication information for the new TDO.

**DesiredAccess:** An access mask that specifies desired access to the TDO handle.

**TrustedDomainHandle:** Used to return the handle for the newly created TDO.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied arguments is invalid. |
| 0xC0000300  STATUS\_NOT\_SUPPORTED\_ON\_SBS | The operation is not supported on a particular product.[<88>](#Appendix_A_88" \o "Product behavior note 88) |
| 0xC00000DD  STATUS\_INVALID\_DOMAIN\_STATE | The operation cannot complete in the current state of the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). |
| 0xC00002B1  STATUS\_DIRECTORY\_SERVICE\_REQUIRED | The [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) was not available on the server. |
| 0xC0000078  STATUS\_INVALID\_SID | The [**security identifier**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4) is not valid. |
| 0xC00002E9  STATUS\_CURRENT\_DOMAIN\_NOT\_ALLOWED | [**Trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) cannot be established with the current domain. |
| 0xC0000035  STATUS\_OBJECT\_NAME\_COLLISION | Another TDO already exists that matches some of the identifying information of the supplied information. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC0000401  STATUS\_PER\_USER\_TRUST\_QUOTA\_EXCEEDED | The caller's quota for the maximum number of TDOs that can be created by control access right Create-Inbound-Trust is exceeded. |
| 0xC0000402  STATUS\_ALL\_USER\_TRUST\_QUOTA\_EXCEEDED | The combined users' quota for the maximum number of TDOs that can be created by control access right Create-Inbound-Trust is exceeded. |

Processing:

This message MUST be processed in an identical manner to [LsarCreateTrustedDomainEx2](#Section_cc86a55db61948fd998a65cca15efeb9), with the following exceptions.

*AuthenticationInformation* is a structure containing authentication information for the trusted domain. The authentication information is not encrypted, which makes this an insecure message to call. As a result, callers SHOULD NOT invoke this message and SHOULD instead call LsarCreateTrustedDomainEx2.

##### LsarCreateTrustedDomain (Opnum 12)

The LsarCreateTrustedDomain method is invoked to create an object of type [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4) in the server's database.

1. NTSTATUS LsarCreateTrustedDomain(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PLSAPR\_TRUST\_INFORMATION TrustedDomainInformation,
4. [in] ACCESS\_MASK DesiredAccess,
5. [out] LSAPR\_HANDLE\* TrustedDomainHandle
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**TrustedDomainInformation:** Information about the new [**trusted domain object (TDO)**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4) to be created.

**DesiredAccess:** An access mask that specifies the desired access to the TDO handle.

**TrustedDomainHandle:** Used to return the handle for the newly created TDO.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied arguments is invalid. |
| 0xC0000300  STATUS\_NOT\_SUPPORTED\_ON\_SBS | The operation is not supported on a particular product.[<89>](#Appendix_A_89" \o "Product behavior note 89) |
| 0xC00002B1  STATUS\_DIRECTORY\_SERVICE\_REQUIRED | The [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) was not available on the server. |
| 0xC0000078  STATUS\_INVALID\_SID | The [**security identifier**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the trusted domain is not valid. |
| 0xC00002E9  STATUS\_CURRENT\_DOMAIN\_NOT\_ALLOWED | [**Trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) cannot be established with the current [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). |
| 0xC0000035  STATUS\_OBJECT\_NAME\_COLLISION | Another TDO already exists that matches some of the identifying information of the supplied information. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC0000401  STATUS\_PER\_USER\_TRUST\_QUOTA\_EXCEEDED | The caller's quota for the maximum number of TDOs that can be created by control access right Create-Inbound-Trust is exceeded. |
| 0xC0000402  STATUS\_ALL\_USER\_TRUST\_QUOTA\_EXCEEDED | The combined users' quota for the maximum number of TDOs that can be created by control access right Create-Inbound-Trust is exceeded. |

Processing:

This message MUST be processed in an identical manner to [LsarCreateTrustedDomainEx](#Section_6817095dd3414d0393d7e9bdca2d3eef) with the following mapping as input parameters.

*PolicyHandle*: Same.

*TrustedDomainInformation*:

* **Name**: Comes from *TrustedDomainInformation*. Name input parameter.
* **FlatName**: Comes from *TrustedDomainInformation*. Name input parameter.
* **SID**: Comes from *TrustedDomainInformation*. Security identifier (SID) input parameter.
* **TrustDirection**: TRUST\_DIRECTION\_OUTBOUND.
* **TrustType**: TRUST\_TYPE\_DOWNLEVEL.
* **TrustAttributes**: 0.

*AuthenticationInformation*: NULL.

*DesiredAccess*: Same.

*TrustedDomainHandle*: Same.

##### LsarQueryInfoTrustedDomain (Opnum 26)

The LsarQueryInfoTrustedDomain method is invoked to retrieve information about the [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4).

1. NTSTATUS LsarQueryInfoTrustedDomain(
2. [in] LSAPR\_HANDLE TrustedDomainHandle,
3. [in] TRUSTED\_INFORMATION\_CLASS InformationClass,
4. [out, switch\_is(InformationClass)]
5. PLSAPR\_TRUSTED\_DOMAIN\_INFO\* TrustedDomainInformation
6. );

**TrustedDomainHandle:** An open trusted domain object handle.

**InformationClass:** One of the [TRUSTED\_INFORMATION\_CLASS](#Section_360691136c3845e8920e17f8ef36f578) values indicating the type of information the caller is interested in.

**TrustedDomainInformation:** Used to return requested information about the trusted domain object.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the arguments supplied to the function was invalid. |
| 0xC0000003  STATUS\_INVALID\_INFO\_CLASS | The *InformationClass* argument is outside the allowed range. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *TrustedDomainHandle* is not a valid handle. |

Processing:

This message takes three arguments:

*TrustedDomainHandle*: An open handle to a trusted domain object. If the handle is not a valid context handle to a trusted domain object or *TrustedDomainHandle*.HandleType does not equal "Trusted Domain", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *TrustedDomainHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03). The following table specifies the RequiredAccess value to use in this access check for each *InformationClass* value, or indicates if no processing is supported, regardless of access granted. There are several methods in the Local Security Authority (Domain Policy) Remote Protocol that query [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4) information. All of them enforce the same rights assignments based on information class as described in the following table.

| Value of InformationClass parameter | RequiredAccess value |
| --- | --- |
| TrustedDomainNameInformation  TrustedDomainInformationBasic  TrustedDomainInformationEx  TrustedDomainInformationEx2Internal | TRUSTED\_QUERY\_DOMAIN\_NAME |
| TrustedControllersInformation | Does not apply: This information class is obsolete and cannot be set or queried. The server MUST return STATUS\_INVALID\_PARAMETER. |
| TrustedPosixOffsetInformation  TrustedDomainSupportedEncryptionTypes | TRUSTED\_QUERY\_POSIX |
| TrustedPasswordInformation  TrustedDomainAuthInformation  TrustedDomainAuthInformationInternal | TRUSTED\_QUERY\_AUTH |
| TrustedDomainFullInformation  TrustedDomainFullInformationInternal  TrustedDomainFullInformation2Internal | TRUSTED\_QUERY\_DOMAIN\_NAME | TRUSTED\_QUERY\_POSIX | TRUSTED\_QUERY\_AUTH |

*InformationClass*: A value from the TRUSTED\_INFORMATION\_CLASS enumeration specifying what type of information the caller is requesting. Not all values are valid. For values outside the TRUSTED\_INFORMATION\_CLASS range, the server MUST reject the request with STATUS\_INVALID\_PARAMETER. Information class values TrustedDomainAuthInformationInternal and TrustedDomainFullInformationInternal MUST be rejected with STATUS\_INVALID\_INFO\_CLASS.

*TrustedDomainInformation*: Used to return the data requested by the caller, in a structure form corresponding to the *InformationClass* parameter. Information MUST be collected from the abstract data model specified in section [3.1.1.5](#Section_0228f75e9725479cb4cd1cacd667343c).

| Value of InformationClass parameter | Information to return |
| --- | --- |
| TrustedDomainNameInformation | Flat Name |
| TrustedPosixOffsetInformation | Posix Offset |
| TrustedDomainInformationEx | Name  Flat Name  Security Identifier  Posix Offset  Trust Type  Trust Direction  Trust Attributes |
| TrustedDomainAuthInformation  TrustedDomainAuthInformationInternal  TrustedDomainFullInformationInternal | Not applicable: This information class cannot be queried. Server MUST return STATUS\_INVALID\_INFO\_CLASS. |
| TrustedDomainFullInformation | Name  Flat Name  Security Identifier  Posix Offset  Trust Type  Trust Direction  Trust Attributes  Trust Incoming and Outgoing Password values MUST be set to 0. |
| TrustedDomainFullInformation2Internall | Name  Flat Name  Security Identifier  Posix Offset  Trust Type  Trust Direction  Trust Attributes  Forest Trust Attributes, as stored in [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) under the msDs-TrustForestTrustInfo attribute ([[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6.9.3).  [**Trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) Incoming and Outgoing Password values MUST be set to 0. |
| TrustedDomainSupportedEncryptionTypes | Supported Encryption Types |
| Other values | Server MUST return STATUS\_INVALID\_PARAMETER. |

If the server is not at DS\_BEHAVIOR\_WIN2003 [**forest functional level**](#gt_b3240417-ca43-4901-90ec-fde55b32b3b8), the presence of the TRUST\_ATTRIBUTE\_FOREST\_TRANSITIVE bit in the **Trust Attributes** field of a trusted domain object MUST NOT be returned by the server.[<90>](#Appendix_A_90" \o "Product behavior note 90)

##### LsarSetInformationTrustedDomain (Opnum 27)

The LsarSetInformationTrustedDomain method is invoked to set information on a [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4).

1. NTSTATUS LsarSetInformationTrustedDomain(
2. [in] LSAPR\_HANDLE TrustedDomainHandle,
3. [in] TRUSTED\_INFORMATION\_CLASS InformationClass,
4. [in, switch\_is(InformationClass)]
5. PLSAPR\_TRUSTED\_DOMAIN\_INFO TrustedDomainInformation
6. );

**TrustedDomainHandle:** A handle to a trusted domain object.

**InformationClass:** A value indicating the type of information requested by the caller.

**TrustedDomainInformation:** Used to supply the information to be set.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the arguments supplied to the function was invalid. |
| 0xC00000DD  STATUS\_INVALID\_DOMAIN\_STATE | The [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) is in the wrong state to perform the stated operation. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *TrustedDomainHandle* is not a valid handle. |

Processing:

This message takes three arguments:

*TrustedDomainHandle*: An open handle to a trusted domain object. If the handle is not a valid context handle to a trusted domain object or *TrustedDomainHandle*.HandleType does not equal "Trusted Domain", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *TrustedDomainHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03). The following table specifies the RequiredAccess value to use in this access check for each *InformationClass* value, or indicates if no processing is supported, regardless of access granted. There are several methods in the Local Security Authority (Domain Policy) Remote Protocol that set [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4) information. All of them enforce the same rights assignments based on information class.

| Value of InformationClass parameter | RequiredAccess value |
| --- | --- |
| TrustedPosixOffsetInformation  TrustedDomainInformationEx | TRUSTED\_SET\_POSIX |
| TrustedDomainFullInformation  TrustedDomainFullInformationInternal | TRUSTED\_SET\_POSIX | TRUSTED\_SET\_AUTH |
| TrustedDomainAuthInformation  TrustedDomainAuthInformationInternal | TRUSTED\_SET\_AUTH |
| TrustedDomainSupportedEncryptionTypes | TRUSTED\_SET\_POSIX |

*InformationClass*: A value from the [TRUSTED\_INFORMATION\_CLASS](#Section_360691136c3845e8920e17f8ef36f578) enumeration specifying what type of information the caller is setting. Not all values are valid. For values outside the TRUSTED\_INFORMATION\_CLASS range, the server MUST reject the request with STATUS\_INVALID\_PARAMETER. Information class values other than the following set SHOULD[<91>](#Appendix_A_91" \o "Product behavior note 91) be rejected with STATUS\_INVALID\_PARAMETER. The set of allowed information class values is:

* TrustedPosixOffsetInformation
* TrustedDomainInformationEx
* TrustedDomainAuthInformation
* TrustedDomainFullInformation
* TrustedDomainAuthInformationInternal
* TrustedDomainFullInformationInternal
* TrustedDomainSupportedEncryptionTypes

*TrustedDomainInformation*: Contains information to be set, appropriate for the *InformationClass* parameter. The server MUST validate the *TrustedDomainInformation* parameter according to information class–specific rules. The rules for internal consistency checking of trusted domain objects are specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6.

Information in the abstract data model specified in section [3.1.1.5](#Section_0228f75e9725479cb4cd1cacd667343c) MUST be updated using *TrustedDomainInformation* and *InformationClass* parameters as follows:

| Value of InformationClass parameter | Information to set |
| --- | --- |
| TrustedPosixOffsetInformation | Posix Offset |
| TrustedDomainInformationEx | Trust Type  Trust Direction  Trust Attributes  Forest Trust Attributes MUST be set to 0 if new [**trust attributes**](#gt_0b8230da-0fd8-492c-b84a-d8467c3dc1ef) do not contain TRUST\_ATTRIBUTE\_FOREST\_TRANSITIVE flag. |
| TrustedDomainAuthInformation  TrustedDomainAuthInformationInternal | Trust Incoming Password  Trust Outgoing Password |
| TrustedDomainFullInformation  TrustedDomainFullInformationInternal | Posix Offset  Trust Type  Trust Direction  Trust Attributes  Trust Incoming Password  Trust Outgoing Password  Forest Trust Attributes MUST be set to 0 if new trust attributes do not contain TRUST\_ATTRIBUTE\_FOREST\_TRANSITIVE flag. |
| TrustedDomainSupportedEncryptionTypes | Supported Encryption Types |
| Other values | Server MUST return STATUS\_INVALID\_PARAMETER. |

The server MUST return STATUS\_INVALID\_DOMAIN\_STATE in the following cases:

* The TrustAttributes TRUST\_ATTRIBUTE\_FOREST\_TRANSITIVE flag is set and the forestFunctionality specified in [MS-ADTS] section 3.1.1.3.2.27 is DS\_BEHAVIOR\_WIN2003 or higher.
* The TrustAttributes TRUST\_ATTRIBUTE\_FOREST\_TRANSITIVE flag is set and the **DnsForestName** and **DnsDomainName** fields in DNS Domain Information (see section [3.1.1.1](#Section_c939c70e8cf04d90a9cc1c5002951723)) do not contain the same value.
* The TrustAttributes TRUST\_ATTRIBUTE\_CROSS\_ORGANIZATION flag is set and the forestFunctionality specified in [MS-ADTS] section 3.1.1.3.2.27 is DS\_BEHAVIOR\_WIN2003 or higher.

If the server is a [**read-only domain controller**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870), it MUST return an error.[<92>](#Appendix_A_92" \o "Product behavior note 92)

If the [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) direction is being set to incoming or bidirectional, then the server MUST create an [**interdomain trust account**](#gt_ba0d31d7-aa03-4e10-936a-a0ebd276ebc9) for this trust, if such an account does not yet exist, and populate it as specified in [MS-ADTS] section 6.1.6.8. The unicodePwd attribute of the account is updated (as specified in [[MS-SAMR]](%5bMS-SAMR%5d.pdf#Section_4df07fab1bbc452f8e927853a3c7e380) section 3.1.1.8.7) with the clear text password (that is, the password value with AuthType being equal to 0x2) in the "Trust Incoming Passwords" information provided.

##### LsarQueryForestTrustInformation (Opnum 73)

The LsarQueryForestTrustInformation method is invoked to retrieve information about a [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) relationship with another [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62).

1. NTSTATUS LsarQueryForestTrustInformation(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PLSA\_UNICODE\_STRING TrustedDomainName,
4. [in] LSA\_FOREST\_TRUST\_RECORD\_TYPE HighestRecordType,
5. [out] PLSA\_FOREST\_TRUST\_INFORMATION\* ForestTrustInfo
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**TrustedDomainName:** The name of the [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4) to query.

**HighestRecordType:** The highest ordinal number of [**forest trust**](#gt_035d9ce5-f117-4251-8d4d-127c462ec4a0) record type that the caller understands.

**ForestTrustInfo:** Used to return the [**forest trust information**](#gt_8c0b82d9-efec-4723-95a9-8564edf9ba44).

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the supplied arguments was invalid. |
| 0xC00000DD  STATUS\_INVALID\_DOMAIN\_STATE | The [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) is in the wrong state of this operation. |
| 0xC00000DF  STATUS\_NO\_SUCH\_DOMAIN | The *TrustedDomainName* is not a recognized domain name. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC0000225  STATUS\_NOT\_FOUND | Forest trust information does not exist for this [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4). |

Processing:

This message takes four arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. *PolicyHandle*.GrantedAccess MUST NOT be considered for this call because the access check MUST happen on the trusted domain object.

*TrustedDomainName*: The name of the trusted domain object to query.

The server MUST return STATUS\_INVALID\_DOMAIN\_STATE if any of the following conditions is TRUE:

* The **DnsForestName** and **DnsDomainName** fields in DNS Domain Information (see section [3.1.1.1](#Section_c939c70e8cf04d90a9cc1c5002951723)) do not contain the same value.
* The forestFunctionality specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.3.2.27 is not DS\_BEHAVIOR\_WIN2003 or higher.
* [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) is not running on this machine.

If a trusted domain object by the name TrustedDomainName does not exist, the server MUST return STATUS\_NO\_SUCH\_DOMAIN.

*HighestRecordType*: The caller sets this argument to the highest [LSA\_FOREST\_TRUST\_RECORD\_TYPE](#Section_700a91e8a1a44e1b9ad6096b3cf0bef0) enum value recognized by the caller. This parameter is ignored by the server.

*ForestTrustInfo*: Used to return the forest trust information associated with the trusted domain object. This corresponds to the Forest Trust Information abstract data model specified in section [3.1.1.5](#Section_0228f75e9725479cb4cd1cacd667343c).

If the trusted domain object is not of the type that supports a forest trust (as determined by the presence or absence of the TRUST\_ATTRIBUTE\_FOREST\_TRANSITIVE attribute), the server MUST return STATUS\_INVALID\_PARAMETER. If the forest trust information does not exist on a trusted domain object that otherwise can support a forest trust, the server MUST return STATUS\_NOT\_FOUND.

The server MUST verify that the caller has access to the trusted domain object as specified in section [3.1.4.2.1](#Section_e5e1e32e4066435db669044fe997eaf7) with *DesiredAccess* set to TRUSTED\_QUERY\_AUTH. There is no method-specific portion of this check.

##### LsarSetForestTrustInformation (Opnum 74)

The LsarSetForestTrustInformation method is invoked to establish a [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) relationship with another [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62) by attaching a set of records called the [**forest trust information**](#gt_8c0b82d9-efec-4723-95a9-8564edf9ba44) to the [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4).

1. NTSTATUS LsarSetForestTrustInformation(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PLSA\_UNICODE\_STRING TrustedDomainName,
4. [in] LSA\_FOREST\_TRUST\_RECORD\_TYPE HighestRecordType,
5. [in] PLSA\_FOREST\_TRUST\_INFORMATION ForestTrustInfo,
6. [in] unsigned char CheckOnly,
7. [out] PLSA\_FOREST\_TRUST\_COLLISION\_INFORMATION\* CollisionInfo
8. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**TrustedDomainName:** The name of the trusted domain object on which to set the forest trust information.

**HighestRecordType:** The highest ordinal [**forest trust**](#gt_035d9ce5-f117-4251-8d4d-127c462ec4a0) record type that the caller understands.

**ForestTrustInfo:** The forest trust information that the caller is trying to set on the trusted domain object.

**CheckOnly:** If not 0, the operation is read-only and does not alter the state of the server's database.

**CollisionInfo:** Used to return information about collisions between different sets of forest trust information in the server's database.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC00000DD  STATUS\_INVALID\_DOMAIN\_STATE | The [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) is not the [**root domain**](#gt_529c766b-af01-4bc8-b853-65fba6c704b3) of the forest, or the forest is not at DS\_BEHAVIOR\_WIN2003 [**forest functional level**](#gt_b3240417-ca43-4901-90ec-fde55b32b3b8). |
| 0xC00000DE  STATUS\_INVALID\_DOMAIN\_ROLE | The server is not the [**primary domain controller**](#gt_663cb13a-8b75-477f-b6e1-bea8f2fba64d). |
| 0xC00000DF  STATUS\_NO\_SUCH\_DOMAIN | The trusted domain object with the name in the *TrustedDomainName* parameter does not exist. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | Some of the parameters supplied were invalid. |

Processing:

This message takes six arguments:

*PolicyHandle*: Open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. *PolicyHandle*.GrantedAccess MUST NOT be considered for this call because the access check MUST happen on the trusted domain object.

*TrustedDomainName*: The name of the trusted domain object to set forest trust information on.

The server MUST return STATUS\_INVALID\_DOMAIN\_STATE if any of the following conditions is TRUE.

* The **DnsForestName** and **DnsDomainName** fields in DNS Domain Information (see section [3.1.1.1](#Section_c939c70e8cf04d90a9cc1c5002951723)) do not contain the same value.
* [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) is not running on this machine.

The server MUST return STATUS\_INVALID\_DOMAIN\_ROLE if the IsEffectiveRoleOwner function specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 3.1.1.5.1.8 returns FALSE with the roleObject parameter set to default NC.

If a trusted domain object by the name *TrustedDomainName* does not exist, the server MUST return STATUS\_NO\_SUCH\_DOMAIN.

The server MUST verify that the caller has access to the trusted domain object as specified in section [3.1.4.2.1](#Section_e5e1e32e4066435db669044fe997eaf7) with *DesiredAccess* set to TRUSTED\_SET\_AUTH. There is no method-specific portion of this check.

The server MUST also make sure that the [**trust attributes**](#gt_0b8230da-0fd8-492c-b84a-d8467c3dc1ef) associated with the trusted domain object referenced by the *TrustedDomainName* parameter has the TRUST\_ATTRIBUTE\_FOREST\_TRANSITIVE set. If the attribute is not present, the server MUST return STATUS\_INVALID\_PARAMETER.

*HighestRecordType*: The caller sets this argument to the highest [LSA\_FOREST\_TRUST\_RECORD\_TYPE](#Section_700a91e8a1a44e1b9ad6096b3cf0bef0) enumeration value recognized by the caller. If this argument is greater in value than the highest record type recognized by the server, the server MUST return STATUS\_INVALID\_PARAMETER.

*ForestTrustInfo*: A collection of forest trust records identifying the topology of the [**trusted forest**](#gt_3b76a71f-9697-4836-9c69-09899b23c21b). The server MUST verify that the forest trust information supplied by the caller is valid by performing a consistency check, as specified in [MS-ADTS] section 6.1.6. Note that "consistent" does not necessarily mean "collision-free". The method for determining collisions is specified in section [3.1.4.7.16.1](#Section_f0e0eefde53c463bb81686fb21931366).

*CheckOnly*: Perform a read-only probing operation. The results will not be persisted in the Local Security Authority (Domain Policy) database, but the set of collision records returned in *CollisionInfo* will be accurate as though the information was persisted.

*CollisionInfo*: A list of collision records. The request is considered successful even if a non-empty set of collisions is returned. The rules for generating collision information are specified in section 3.1.4.7.16.1.

The server MUST store the generated *ForestTrustInfo* in the Forest Trust Information attribute specified in section [3.1.1.5](#Section_0228f75e9725479cb4cd1cacd667343c).

###### Forest Trust Collision Generation

This section describes the rules that the server MUST follow to compute a set of collisions when setting [**forest trust information**](#gt_8c0b82d9-efec-4723-95a9-8564edf9ba44) on a [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4).

Forest trust information across all [**trusted forests**](#gt_3b76a71f-9697-4836-9c69-09899b23c21b) is always internally consistent. This is an invariant that the server MUST enforce. When new forest trust information is added to the server's policy database, the server MUST ensure that the overall forest trust information remains consistent. The server does so by disabling the entries in the new forest trust information structure that would violate this internal consistency. The server communicates the entries that are inconsistent with existing forest trust information back to the client by computing and returning a set of "collision entries".

The rules that govern consistency of forest trust information are specified in [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6 and are listed here for convenience. To be exact, there are two sets of rules, one for top-level name entries, and one for [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) information entries.

The rules for top-level name entries are as follows:

* An enabled (that is, non-conflict) top-level name record must not be equal to an enabled top-level name for another trusted domain object or to any of the DNS tree names within the current [**forest**](#gt_fd104241-4fb3-457c-b2c4-e0c18bb20b62). Equality is computed using case-insensitive string comparison. If the strings differ only by one trailing '.' character, the difference is ignored.
* The top-level name must not be subordinate to an enabled top-level name for another trusted domain object, unless the other trusted domain object has a corresponding exclusion record.
* A top-level name must not be superior to an enabled top-level name for another trusted domain object, unless the current trusted domain object has a corresponding exclusion record.

If any of these rules are violated, a top-level name is considered in conflict. In this case, a collision record is generated with the following values:

**Index**: Ordinal number of a [**forest trust**](#gt_035d9ce5-f117-4251-8d4d-127c462ec4a0) record supplied by the caller that generated the collision.

**Type**: CollisionTdo or CollisionXref, depending on whether the collision was caused by an external-to-forest domain or an internal-to-forest domain.

**Flags**: LSA\_TLN\_DISABLED\_CONFLICT

**Name**: [**DNS name**](#gt_102a36e2-f66f-49e2-bee3-558736b2ecd5) of the TDO that contained the forest trust information with which this entry has collided.

The rules for domain information entries are as follows:

* + The [**security identifier**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of this entry must not be equal to that of an enabled domain information entry belonging to a different forest or any of the domains that comprise the current forest.
  + The NetBIOS name of this entry must not be claimed by any other forest with which this forest has a [**trust**](#gt_5ee032d0-d944-4acb-bbb5-b1cfc7df6db6) relationship or by any domain within the current forest.
  + The DNS name of this entry must not be claimed by any other forest with which this forest has a trust relationship or by the current forest.

If any of these rules are violated, a domain information entry is considered to be in conflict. In this case, a collision record is generated with the following values:

**Index**: Ordinal number of a forest trust record supplied by the caller that generated the collision.

**Type**: CollisionTdo or CollisionXref, depending on whether the collision was caused by an external-to-forest or internal-to-forest domain.

**Flags**: LSA\_SID\_DISABLED\_CONFLICT if the collision was caused by a security identifier component of the record. LSA\_NB\_DISABLED\_CONFLICT if the collision was caused by a NetBIOS name component of the record.

Entries that have been disabled by administrative action or through conflict are not considered in computing consistency checks.

#### Privilege Methods

The message processing of methods in this section MUST use the abstract data model specified in section [3.1.1.2.1](#Section_1a92af76d45f42c3b67cf1dc61bd6ee1).

| Method (opnum) | Summary |
| --- | --- |
| LsarEnumeratePrivileges (opnum 2) | Enumerates all [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) known to the server. |
| LsarLookupPrivilegeValue (opnum 31) | Maps the well-known name of a privilege into the server-specific [**locally unique identifier (LUID)**](#gt_96b64af9-1896-4bde-b988-54d469c5affd). |
| LsarLookupPrivilegeName (opnum 32) | Maps the server-specific LUID of a privilege into a well-known privilege name. |
| LsarLookupPrivilegeDisplayName (opnum 33) | Maps the well-known name of a privilege into a human-readable name in the caller's language. |

##### LsarEnumeratePrivileges (Opnum 2)

The LsarEnumeratePrivileges method is invoked to enumerate all [**privileges**](#gt_d8092e10-b227-4b44-b015-511bb8178940) known to the system. This method can be called multiple times to return its output in fragments.

1. NTSTATUS LsarEnumeratePrivileges(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in, out] unsigned long\* EnumerationContext,
4. [out] PLSAPR\_PRIVILEGE\_ENUM\_BUFFER EnumerationBuffer,
5. [in] unsigned long PreferedMaximumLength
6. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**EnumerationContext:** A pointer to a context value that is used to resume enumeration, if necessary.

**EnumerationBuffer:** A pointer to a structure that will contain the results of the enumeration.

**PreferedMaximumLength:** A value that indicates the approximate size of the data to be returned.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0x00000105  STATUS\_MORE\_ENTRIES | More information is available to successive calls. |
| 0x8000001A  STATUS\_NO\_MORE\_ENTRIES | No more entries are available from the enumeration. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the parameters supplied was invalid. This can happen if *EnumerationBuffer* is NULL or *EnumerationContext* is NULL. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This method takes four arguments:

*PolicyHandle*: Open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to POLICY\_VIEW\_LOCAL\_INFORMATION.

*EnumerationContext*: A number that indicates a starting index at which to begin the enumeration. The server MUST always return all privileges in the same order, starting at the object whose index is *EnumerationContext*. To initiate a new enumeration, the client sets *EnumerationContext* to zero; otherwise, the client sets *EnumerationContext* to a value returned by a previous call to the method.

The server MUST return STATUS\_INVALID\_PARAMETER if the *EnumerationContext* parameter is NULL.

*EnumerationBuffer*: Used to return the results of enumeration. The server MUST fill *EnumerationBuffer* with as many privilege objects as possible, as determined by *PreferedMaximumLength*. If the size of all remaining objects is less than or equal to *PreferedMaximumLength*, the server MUST fill *EnumerationBuffer* with all objects. If the size of all remaining objects is greater than *PreferedMaximumLength*, the server MUST fill *EnumerationBuffer* with objects such that the size of the privilege objects returned is greater than or equal to *PreferedMaximumLength*, but would be less than *PreferedMaximumLength* if the last object had not been added to *EnumerationBuffer*. If there are no more objects than are returned in *EnumerationBuffer*, the server MUST return STATUS\_NO\_MORE\_ENTRIES. If there are more database objects than are returned in *EnumerationBuffer*, the server MUST set the *EnumerationContext* value to the index value that would allow it to resume enumeration correctly when this method is called again, and the server MUST return STATUS\_MORE\_ENTRIES. Note that this return value is not an error status.

*PreferedMaximumLength*: An indication about the approximate size, in bytes, of the data to return. Any unsigned 32-bit value is valid for the *PreferedMaximumLength* parameter.

##### LsarLookupPrivilegeValue (Opnum 31)

The LsarLookupPrivilegeValue method is invoked to map the name of a [**privilege**](#gt_d8092e10-b227-4b44-b015-511bb8178940) into a [**locally unique identifier (LUID)**](#gt_96b64af9-1896-4bde-b988-54d469c5affd) by which the privilege is known on the server. The locally unique value of the privilege can then be used in subsequent calls to other methods, such as [LsarAddPrivilegesToAccount](#Section_8a542f26243d4341ada58fed194bfcf8).

1. NTSTATUS LsarLookupPrivilegeValue(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_UNICODE\_STRING Name,
4. [out] PLUID Value
5. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**Name:** A string containing the name of a privilege.

**Value:** Used to return a LUID assigned by the server to the privilege by this name.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One or more of the supplied parameters was invalid. |
| 0xC0000060  STATUS\_NO\_SUCH\_PRIVILEGE | The privilege name is not recognized by the server. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message takes three arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to POLICY\_LOOKUP\_NAMES.

*Name*: The string name of the privilege.

*Value*: Used to return the LUID corresponding to the *Name* argument.

If the value in the *Name* argument is not recognized by the server, the server MUST fail the request with STATUS\_NO\_SUCH\_PRIVILEGE. The privileges recognized by the server are specified in section [3.1.1.2.1](#Section_1a92af76d45f42c3b67cf1dc61bd6ee1).

##### LsarLookupPrivilegeName (Opnum 32)

The LsarLookupPrivilegeName method is invoked to map the [**LUID**](#gt_96b64af9-1896-4bde-b988-54d469c5affd) of a [**privilege**](#gt_d8092e10-b227-4b44-b015-511bb8178940) into a string name by which the privilege is known on the server.

1. NTSTATUS LsarLookupPrivilegeName(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PLUID Value,
4. [out] PRPC\_UNICODE\_STRING\* Name
5. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**Value:** A LUID that the caller wishes to map to a string name.

**Name:** Used to return the string name corresponding to the supplied LUID.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC0000060  STATUS\_NO\_SUCH\_PRIVILEGE | The supplied LUID is not recognized by the server. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This message takes three arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to POLICY\_LOOKUP\_NAMES.

*Value*: The LUID of the privilege.

*Name*: Used to return the name corresponding to the LUID contained in the *Value* argument.

If the LUID in the *Value* argument is not recognized by the server, the server MUST fail the request with STATUS\_NO\_SUCH\_PRIVILEGE. The privileges recognized by the server are specified in section [3.1.1.2.1](#Section_1a92af76d45f42c3b67cf1dc61bd6ee1).

##### LsarLookupPrivilegeDisplayName (Opnum 33)

The LsarLookupPrivilegeDisplayName method is invoked to map the name of a [**privilege**](#gt_d8092e10-b227-4b44-b015-511bb8178940) into a display text string in the caller's language.

1. NTSTATUS LsarLookupPrivilegeDisplayName(
2. [in] LSAPR\_HANDLE PolicyHandle,
3. [in] PRPC\_UNICODE\_STRING Name,
4. [in] short ClientLanguage,
5. [in] short ClientSystemDefaultLanguage,
6. [out] PRPC\_UNICODE\_STRING\* DisplayName,
7. [out] unsigned short\* LanguageReturned
8. );

**PolicyHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle obtained from either [LsarOpenPolicy](#Section_2a482ccf1f8946938594855ff738ae8a) or [LsarOpenPolicy2](#Section_9456a9637c214710af77d0a2f5a72d6b).

**Name:** A string containing the name of a privilege.

**ClientLanguage:** An identifier of the client's language.

**ClientSystemDefaultLanguage:** An identifier of the default language of the caller's machine.

**DisplayName:** Used to return the display name of the privilege in the language pointed to by the *LanguageReturned* value.

**LanguageReturned:** An identifier of the language in which *DisplayName* was returned.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One or more of the supplied parameters was invalid. |
| 0xC0000060  STATUS\_NO\_SUCH\_PRIVILEGE | The supplied [**LUID**](#gt_96b64af9-1896-4bde-b988-54d469c5affd) is not recognized by the server. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *PolicyHandle* is not a valid handle. |

Processing:

This method takes six arguments:

*PolicyHandle*: An open handle to the policy object. If the handle is not a valid context handle to the policy object or *PolicyHandle*.HandleType does not equal "Policy", the server MUST return STATUS\_INVALID\_HANDLE. The server MUST verify that *PolicyHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to POLICY\_LOOKUP\_NAMES.

*Name*: A string name of the privilege. The server MUST attempt to locate the entry with the same name in the data store specified in section [3.1.1.2.1](#Section_1a92af76d45f42c3b67cf1dc61bd6ee1). If the entry cannot be located, the server MUST return STATUS\_NO\_SUCH\_PRIVILEGE.

*ClientLanguage*: A numerical identifier of the language in which the caller wishes to receive the display name. The server MUST try to locate the privilege description in the language that is identified by this parameter. If the data store does not have this language, the server MUST try the next parameter.

*ClientSystemDefaultLanguage*: An identifier of the default language of the caller. This might be different than the *ClientLanguage* parameter. If the data store does not have the description in the previous language, the server MUST try to find the description in this language.

*DisplayName*: Used to return the description of the privilege. If neither *ClientLanguage* nor *ClientSystemDefaultLanguage* can be found, the server MUST return the description in the server's own language.

*LanguageReturned*: Used to return the language ID of *DisplayName*. This might be different from the language ID that was requested.

#### Common Object Methods

The message processing of methods in this section MUST use the abstract data model defined in section [3.1.1](#Section_0877fdc4184f40b0b378f50d6647d23e).

| Method (Opnum) | Summary |
| --- | --- |
| LsarQuerySecurityObject (opnum 3) | Retrieves the [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) associated with an object. |
| LsarSetSecurityObject (opnum 4) | Sets a security descriptor on an object. |
| LsarDeleteObject (opnum 34) | Deletes an object from the policy database. |
| LsarClose (opnum 0) | Closes an open handle. |

##### LsarQuerySecurityObject (Opnum 3)

The LsarQuerySecurityObject method is invoked to query security information that is assigned to a database object. It returns the [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) of the object.

1. NTSTATUS LsarQuerySecurityObject(
2. [in] LSAPR\_HANDLE ObjectHandle,
3. [in] SECURITY\_INFORMATION SecurityInformation,
4. [out] PLSAPR\_SR\_SECURITY\_DESCRIPTOR\* SecurityDescriptor
5. );

**ObjectHandle:** An open object handle of any type.

**SecurityInformation:** A bitmask specifying which portions of the security descriptor the caller is interested in.

**SecurityDescriptor:** Used to return the security descriptor containing the elements requested by the caller.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC00000BB  STATUS\_NOT\_SUPPORTED | The request is not supported. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *ObjectHandle* is not a valid handle. |

Processing:

This message takes three arguments:

*ObjectHandle*: Can be an open handle of any type. If the handle is not a valid context handle to an object or *ObjectHandle*.PolicyType is not one of the following:

* "Policy" for handles to policy objects
* "Account" for handles to [**account objects**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b)
* "Secret" for handles to [**secret objects**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d)
* "Trusted Domain" for handles to [**trusted domain objects**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4)

The server MUST return STATUS\_INVALID\_HANDLE. The access required for a successful completion of this request depends on the *SecurityInformation* parameter. The server MUST verify that *ObjectHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03). The following pseudocode specifies the RequiredAccess value to use in this access check.

1. Set RequiredAccess equal to 0
2. IF ((SecurityInformation & OWNER\_SECURITY\_INFORMATION) || (SecurityInformation & GROUP\_SECURITY\_INFORMATION) || (SecurityInformation & DACL\_SECURITY\_INFORMATION)) THEN
3. RequiredAccess |= READ\_CONTROL
4. END IF
5. IF (SecurityInformation & SACL\_SECURITY\_INFORMATION) THEN
6. RequiredAccess |= ACCESS\_SYSTEM\_SECURITY
7. END IF

*SecurityInformation*: A set of bits specifying which portions of the security descriptor the caller is interested in retrieving. The various bits and the associated access rights are specified in section [2.2.1.3](#Section_62175da4e30f4c12b1c4dae0434e38af).

*SecurityDescriptor*: An output parameter. If access checks pass, the server MUST fill this information with a valid self-relative security descriptor containing only the fields requested by the caller. The server MUST NOT put information into the security descriptor that the caller did not request.

It is valid for the server to not support this method for all object types. If an object does not support this method, the server MUST return STATUS\_NOT\_SUPPORTED.[<93>](#Appendix_A_93" \o "Product behavior note 93)

##### LsarSetSecurityObject (Opnum 4)

The LsarSetSecurityObject method is invoked to set a [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) on an object.

1. NTSTATUS LsarSetSecurityObject(
2. [in] LSAPR\_HANDLE ObjectHandle,
3. [in] SECURITY\_INFORMATION SecurityInformation,
4. [in] PLSAPR\_SR\_SECURITY\_DESCRIPTOR SecurityDescriptor
5. );

**ObjectHandle:** An open handle to an existing object.

**SecurityInformation:** A bitmask specifying which portions of the security descriptor are to be set.

**SecurityDescriptor:** The security descriptor to be set.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC000009A  STATUS\_INSUFFICIENT\_RESOURCES | There are insufficient resources to complete the request. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC0000079  STATUS\_INVALID\_SECURITY\_DESCR | The supplied security descriptor is invalid. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One of the parameters supplied was invalid. For instance, *SecurityDescriptor* is NULL. |
| 0xC00000BB  STATUS\_NOT\_SUPPORTED | The operation is not supported for this object. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *ObjectHandle* is not a valid handle. |

Processing:

This message takes three arguments:

*ObjectHandle*: Can be an open handle of any type. If the handle is not a valid context handle to an object or *ObjectHandle*.PolicyType is not one of the following:

* "Policy" for handles to policy objects
* "Account" for handles to [**account objects**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b)
* "Secret" for handles to [**secret objects**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d)
* "Trusted Domain" for handles to [**trusted domain objects**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4)

The server MUST return STATUS\_INVALID\_HANDLE. The access required for a successful completion of this request depends on the *SecurityInformation* parameter. The server MUST verify that *ObjectHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03). The following pseudocode specifies the RequiredAccess value to use in this access check.

1. Set RequiredAccess equal to 0
2. IF ((SecurityInformation & OWNER\_SECURITY\_INFORMATION) || (SecurityInformation & GROUP\_SECURITY\_INFORMATION) || (SecurityInformation & DACL\_SECURITY\_INFORMATION)) THEN
3. RequiredAccess |= READ\_CONTROL
4. END IF
5. IF (SecurityInformation & SACL\_SECURITY\_INFORMATION) THEN
6. RequiredAccess |= ACCESS\_SYSTEM\_SECURITY
7. END IF

*SecurityInformation*: A set of bits specifying which portions of the security descriptor the caller is interested in setting. The various bits and the associated access rights are specified in section [2.2.1.3](#Section_62175da4e30f4c12b1c4dae0434e38af).

*SecurityDescriptor*: Expects a valid self-relative security descriptor that the caller is trying to set. If this security descriptor is invalid, the server MUST return the STATUS\_INVALID\_SECURITY\_DESCR status code. If the security descriptor is NULL, the server MUST return STATUS\_INVALID\_PARAMETER.

It is valid for the server to not support this method for all object types.[<94>](#Appendix_A_94" \o "Product behavior note 94)

The server MUST return STATUS\_INSUFFICIENT\_RESOURCES if it runs out of memory while servicing the request.

##### LsarDeleteObject (Opnum 34)

The LsarDeleteObject method is invoked to delete an open [**account object**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b), [**secret object**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d), or [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4).

1. NTSTATUS LsarDeleteObject(
2. [in, out] LSAPR\_HANDLE\* ObjectHandle
3. );

**ObjectHandle:** A handle to an open object of the correct type to be deleted. After successful completion of the call, the handle value cannot be reused.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000022  STATUS\_ACCESS\_DENIED | The caller does not have the permissions to perform this operation. |
| 0xC000000D  STATUS\_INVALID\_PARAMETER | One or more of the supplied parameters was invalid. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *ObjectHandle* is not a valid handle. |

Processing:

This message takes one input parameter.

*ObjectHandle*: An open handle to an object that is to be deleted. If the handle is not a valid context handle to an object or *ObjectHandle*.PolicyType is not one of the following:

* "Policy" for handles to policy objects
* "Account" for handles to account objects
* "Secret" for handles to secret objects
* "Trusted Domain" for handles to trusted domain objects

The server MUST return STATUS\_INVALID\_HANDLE. Policy objects cannot be deleted. Attempts to delete policy objects MUST fail with STATUS\_INVALID\_PARAMETER. For other object types, the server MUST verify that *ObjectHandle* grants access as specified in section [3.1.4.2.2](#Section_d1c2802a70d54f81843c6523ab0c5e03) with RequiredAccess set to DELETE.

The server MUST make all subsequent requests to deleted objects through already opened handles fail with STATUS\_INVALID\_HANDLE. The deleted handle MUST be automatically closed by the server; the caller need not close it.

If the object being deleted is a [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4), then the server MUST also check whether an [**interdomain trust account**](#gt_ba0d31d7-aa03-4e10-936a-a0ebd276ebc9) with name "<Trusted Domain NetBIOS Name>$" exists. If it exists, the server MUST delete that account along with the trusted domain.

The server MUST free any resources associated with the LsaContextHandle element (section [3.1.1.7](#Section_1011130B0413460D81EDD1821D141639)) that is represented by *ObjectHandle*, as specified in section [3.1.6.1](#Section_2184a151ce744e1e8ba725384f51dbf8), LSAPR\_HANDLE\_rundown.

The fact that a handle is no longer usable is communicated to the [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a) by returning a NULL value in the handle parameter, as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 5.1.6.

##### LsarClose (Opnum 0)

The LsarClose method frees the resources held by a context handle that was opened earlier. After response, the context handle will no longer be usable, and any subsequent uses of this handle will fail.

1. NTSTATUS LsarClose(
2. [in, out] LSAPR\_HANDLE\* ObjectHandle
3. );

**ObjectHandle:** The context handle to be freed. On response, it MUST be set to 0.

**Return Values:** The following is a summary of the return values that an implementation MUST return, as specified by the message processing that follows.

| Return value/code | Description |
| --- | --- |
| 0x00000000  STATUS\_SUCCESS | The request was successfully completed. |
| 0xC0000008  STATUS\_INVALID\_HANDLE | *ObjectHandle* is not a valid handle. |

Processing:

A handle of any type can be closed by calling LsarClose. Successful calls to [LsarDeleteObject](#Section_8d0aa2dc22b64bc3b5d279b4b0ad7bce), which deletes an object to which the caller has an open handle, will also close the handle.

If *ObjectHandle* is invalid, the server MUST return STATUS\_INVALID\_HANDLE.

The server MUST free any resources associated with the LsaContextHandle element (section [3.1.1.7](#Section_1011130B0413460D81EDD1821D141639)) that is represented by *ObjectHandle*, as specified in section [3.1.6.1](#Section_2184a151ce744e1e8ba725384f51dbf8), LSAPR\_HANDLE\_rundown.

The fact that a handle is closed is communicated to the [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a) by returning a NULL value in the handle parameter, as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 5.1.6.

Closing one handle MUST NOT affect any other handle on the server; that is, handles obtained using a policy handle MUST continue to be valid after that policy handle is closed.

#### Data Validation

Data types defined in section [2.2](#Section_4d4678cf32154ecc8dc95a2aaa0e1eb0) are subject to a set of validation rules, in addition to any already noted. For structures that contain other structures or sets of other structures, the validation for those structures MUST be enforced when validating the containing structure. All constraints in the following tables MUST be satisfied; on failure, an error NTSTATUS code MUST be returned.

| Data type | Validations |
| --- | --- |
| LSA\_UNICODE\_STRING  RPC\_UNICODE\_STRING  LSAPR\_CR\_CIPHER\_VALUE | * Length MUST be a multiple of 2.[<95>](#Appendix_A_95" \o "Product behavior note 95) * Length MUST be less than or equal to MaximumLength. * If Length is not 0, Buffer MUST NOT be NULL. * The **Buffer** field MUST NOT contain any NULL Unicode characters in the first **Length** bytes.[<96>](#Appendix_A_96" \o "Product behavior note 96) |
| RPC\_SID | * Revision MUST be 1. * SubAuthorityCount MUST be less than or equal to 15.   Additionally, if the [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) is a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) SID:   * IdentifierAuthority MUST be {0,0,0,0,0,5}. * SubAuthorityCount MUST be greater than 3. * SubAuthority[0] MUST be 0x15. |
| LSAPR\_SR\_SECURITY\_DESCRIPTOR | * Revision MUST be 1. * The [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) must conform to the definition for self-relative security descriptor in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.6. |
| LSAPR\_LUID\_AND\_ATTRIBUTES | * Luid.HighPart SHOULD NOT be 0.[<97>](#Appendix_A_97" \o "Product behavior note 97) * Luid.LowPart SHOULD be less than or equal to 35.[<98>](#Appendix_A_98" \o "Product behavior note 98) * Attributes SHOULD have only combinations of bits (0x00000001 & 0x00000002) set.[<99>](#Appendix_A_99" \o "Product behavior note 99) |
| LSAPR\_PRIVILEGE\_SET | * If PrivilegeCount is not 0, Privilege MUST NOT be NULL. * Each Privilege MUST pass validation for LSAPR\_LUID\_AND\_ATTRIBUTES. * There MUST be no duplicate elements in the Privilege array. |
| LSAPR\_OBJECT\_ATTRIBUTES | RootDirectory MUST be NULL. |
| ACCESS\_MASK | SHOULD conform to the defined bits for ACCESS\_MASK. |
| POLICY\_INFORMATION\_CLASS | MUST be greater than or equal to one and MUST be less than or equal to 14, which corresponds to the value: PolicyLocalAccountDomainInformation. |
| POLICY\_AUDIT\_LOG\_INFO | No additional validation. |
| LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO | * MaximumAuditEventCount MUST NOT be 0. * MaximumAuditEventCount MUST be less than or equal to 8. * EventAuditingOptions MUST NOT be NULL. * EventAuditingOptions and 0xFFFFFFF8 MUST be 0. |
| LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO | * DomainName MUST satisfy RPC\_UNICODE\_STRING validations. * DomainSid MUST satisfy RPC\_SID validations, including those for domain SIDs. |
| LSAPR\_POLICY\_PRIMARY\_DOM\_INFO | * Name MUST satisfy RPC\_UNICODE\_STRING validations. * Name.Length MUST be less than or equal 30. * SID MUST either be NULL or satisfy RPC\_SID validations, including those for domain SIDs. |
| LSAPR\_POLICY\_DNS\_DOMAIN\_INFO | * Name MUST pass RPC\_UNICODE\_STRING validations. * Name.Length MUST be less than or equal to 30. * DnsDomainName MUST satisfy RPC\_UNICODE\_STRING validations. * DnsForestName MUST satisfy RPC\_UNICODE\_STRING validations. * SID MUST either be NULL or satisfy RPC\_SID validations, including those for domain SID. |
| LSAPR\_POLICY\_PD\_ACCOUNT\_INFO | Name MUST satisfy RPC\_UNICODE\_STRING validations. |
| POLICY\_LSA\_SERVER\_ROLE\_INFO | LsaServerRole MUST be 2 OR 3. |
| LSAPR\_CR\_CIPHER\_VALUE | MaximumLength MUST be greater than or equal to Length. |
| LSAPR\_POLICY\_REPLICA\_SRCE\_INFO | * ReplicaSource MUST satisfy RPC\_UNICODE\_STRING validation. * ReplicaAccountName must satisfy RPC\_UNICODE\_STRING validation. |
| POLICY\_MODIFICATION\_INFO | ModifiedId MUST not be 0. |
| POLICY\_AUDIT\_FULL\_SET\_INFO | No validation. |
| LSAPR\_POLICY\_DOMAIN\_EFS\_INFO | If InfoLength is not 0, EfsBlob MUST NOT be NULL. |
| TRUSTED\_INFORMATION\_CLASS | MUST be greater than or equal to 1 and less than or equal to 13. |
| LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION | * If IncomingAuthInfos is not 0, IncomingAuthenticationInformation MUST NOT be NULL. * IncomingAuthInfos MUST be 0 or 1. * If OutgoingAuthInfos is not 0, OutgoingAuthenticationInformation MUST NOT be NULL. * OutgoingAuthInfos MUST be 0 or 1. * Each IncomingPreviousAuthenticationInformation MUST satisfy validation for LSAPR\_AUTH\_INFORMATION. * Each IncomingAuthenticationInformation MUST satisfy validation for LSAPR\_AUTH\_INFORMATION. * Each OutgoingPreviousAuthenticationInformation MUST satisfy validation for LSAPR\_AUTH\_INFORMATION. * Each OutgoingAuthenticationInformation MUST satisfy validation for LSAPR\_AUTH\_INFORMATION. |
| LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION | * Information MUST satisfy LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX validation. * AuthInformation MUST satisfy LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION validation. |
| LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2 | * Information MUST satisfy LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 validation. * FlatName MUST satisfy RPC\_UNICODE\_STRING validation. * SID MUST be NULL or satisfy RPC\_SID validation, including domain SID validation. * If ForestTrustLength is not 0, ForestTrustInfo MUST NOT be NULL. |
| LSAPR\_AUTH\_INFORMATION | If AuthInfoLength is not 0, AuthInfo MUST NOT be NULL. |
| LSA\_FOREST\_TRUST\_DOMAIN\_INFO | * SID MUST satisfy RPC\_SID validation, including domain SID validation. * DnsName MUST satisfy RPC\_UNICODE\_STRING validation. * NetbiosName MUST satisfy RPC\_UNICODE\_STRING validation. |
| LSA\_FOREST\_TRUST\_BINARY\_DATA | If Length is not 0, Buffer MUST NOT be NULL. |
| LSA\_FOREST\_TRUST\_RECORD | * For ForestTrustType = ForestTrustTopLevelName or ForestTrustTopLevelNameEx, ForestTrustData.TopLevelName MUST satisfy RPC\_UNICODE\_STRING validation. * For ForestTrustType = ForestTrustDomainInfo, ForestTrustData.DomainInfo MUST satisfy LSA\_FOREST\_TRUST\_DOMAIN\_INFO validation. |
| LSA\_FOREST\_TRUST\_INFORMATION | * If RecordCount is not 0, Entries MUST NOT be NULL. * Each one of Entries MUST satisfy LSA\_FOREST\_TRUST\_RECORD validation. |
| LSA\_FOREST\_TRUST\_COLLISION\_RECORD | Name MUST satisfy RPC\_UNICODE\_STRING validation. |
| LSA\_FOREST\_TRUST\_COLLISION\_INFORMATION | * If RecordCount is not 0, Entries MUST NOT be NULL. * Each one of Entries MUST satisfy LSA\_FOREST\_TRUST\_COLLISION\_RECORD validation. |
| LSAPR\_HANDLE | MUST not be NULL. |
| LSAPR\_ACCOUNT\_INFORMATION | SID MUST satisfy RPC\_SID validation. |
| LSAPR\_ACCOUNT\_ENUM\_BUFFER | * If EntriesRead is not 0, Information MUST NOT be NULL. * Each Information element MUST satisfy LSAPR\_ACCOUNT\_INFORMATION validation. |
| LSAPR\_POLICY\_PRIVILEGE\_DEF | Name MUST satisfy RPC\_UNICODE\_STRING validation. |
| LSAPR\_PRIVILEGE\_ENUM\_BUFFER | * If Entries is not 0, Privileges MUST NOT be NULL. * Each element in Entries MUST satisfy LSAPR\_POLICY\_PRIVILEGE\_DEF validation. |
| LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_BASIC | * Name MUST satisfy RPC\_UNICODE\_STRING validation. * SID MUST be NULL or MUST satisfy RPC\_SID validation including domain SID validation. |
| LSAPR\_TRUSTED\_ENUM\_BUFFER | * If EntriesRead is not 0, Information MUST NOT be NULL. * Each element in Information MUST satisfy LSAPR\_TRUST\_INFORMATION validation. |
| LSAPR\_TRUSTED\_PASSWORD\_INFO | OldPassword and Password MUST satisfy LSAPR\_CR\_CIPHER\_VALUE validation. |
| LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO | Name MUST satisfy RPC\_UNICODE\_STRING validation. |
| LSAPR\_USER\_RIGHT\_SET | * If Entries is not 0, UserRights MUST NOT be NULL. * Each element in UserRights MUST satisfy RPC\_UNICODE\_STRING validation. |
| LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX | * Name MUST satisfy RPC\_UNICODE\_STRING validation. * FlatName MUST satisfy RPC\_UNICODE\_STRING validation. * SID MUST be NULL or MUST satisfy RPC\_SID validation including domain SID validation. |

### Timer Events

No protocol timer events are required on the [**RPC server**](#gt_ae65dac0-cd24-4e83-a946-6d1097b71553) other than the timers required in the underlying [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a).

### Other Local Events

No additional local events are used on the [**RPC server**](#gt_ae65dac0-cd24-4e83-a946-6d1097b71553) other than the events maintained in the underlying [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a).

#### LSAPR\_HANDLE\_rundown

This function implements the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle rundown routine for the LsaContextHandle context handle type (section [3.1.1.7](#Section_1011130B0413460D81EDD1821D141639)). When invoked, the LSAPR\_HANDLE\_rundown procedure MUST free all server resources associated with LsaContextHandle.Object. The server MUST then set LsaContextHandle.Object to 0.

For more information, see [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 5.1.6. An implementation of this protocol SHOULD use this functionality.

# Protocol Examples

The following sections describe several common scenarios from the client's perspective to illustrate the function of the Local Security Authority (Domain Policy) Remote Protocol. "Send" implies that the direction is from client to server, and "Receive" implies the opposite direction.

## Manipulating Account Objects

This section illustrates a message exchange pertaining to [**account objects**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b).

1. Message 1: Open the policy object.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarOpenPolicy2 | SystemName | "Arbitrary String" |
| Send  LsarOpenPolicy2 | ObjectAttributes | Ignored, except for the **RootDirectory** field, which is NULL. |
| Send  LsarOpenPolicy2 | DesiredAccess | POLICY\_VIEW\_LOCAL\_INFORMATION | POLICY\_CREATE\_ACCOUNT | POLICY\_LOOKUP\_NAMES |

1. Message 2: Success; return the policy object handle.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarOpenPolicy2 | Status | STATUS\_SUCCESS |
| Receive  LsarOpenPolicy2 | PolicyHandle | [Implementation-specific value] |

1. Message 3: Attempt to create an account object with [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) S-1-5-21-123-123-123-1005.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarCreateAccount | PolicyHandle | [Implementation-specific value returned in Step 2.] |
| Send  LsarCreateAccount | AccountSid | "S-1-5-21-123-123-123-1005" |
| Send  LsarCreateAccount | DesiredAccess | READ\_CONTROL | WRITE\_DAC | ACCOUNT\_ADJUST\_PRIVILEGES | ACCOUNT\_ADJUST\_SYSTEM\_ACCESS | ACCOUNT\_VIEW |

1. Message 4: Failure: Account already exists.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarCreateAccount | Status | STATUS\_OBJECT\_NAME\_COLLISION |
| Receive  LsarCreateAccount | AccountHandle | NULL |

1. Message 5: Attempt to open the account object with SID S-1-5-21-123-123-123-1005.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarOpenAccount | PolicyHandle | [Implementation-specific value] |
| Send  LsarOpenAccount | AccountSid | "S-1-5-21-123-123-123-1005" |
| Send  LsarOpenAccount | DesiredAccess | READ\_CONTROL | WRITE\_DAC | ACCOUNT\_ADJUST\_PRIVILEGES | ACCOUNT\_ADJUST\_SYSTEM\_ACCESS | ACCOUNT\_VIEW |

1. Message 6: Success: Return the account object handle.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarOpenAccount | Status | STATUS\_SUCCESS |
| Receive  LsarOpenAccount | AccountHandle | [Implementation-specific value] |

1. Message 7: Retrieve the [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) of the account object.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarQuerySecurityObject | ObjectHandle | [Implementation-specific value returned in Step 6.] |
| Send  LsarQuerySecurityObject | SecurityInformation | DACL\_SECURITY\_INFORMATION |

1. Message 8: Success: Return the security descriptor.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarQuerySecurityObject | Status | STATUS\_SUCCESS |
| Receive  LsarQuerySecurityObject | SecurityDescriptor | Security descriptor of the account object in self-relative form. |

1. Message 9: Update the [**discretionary access control list (DACL)**](#gt_d727f612-7a45-48e4-9d87-71735d62b321) on the account object.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarSetSecurityObject | ObjectHandle | [Implementation-specific value returned in Step 6.] |
| Send  LsarSetSecurityObject | SecurityInformation | DACL\_SECURITY\_INFORMATION |
| Send  LsarSetSecurityObject | SecurityDescriptor | Security descriptor representation of the DACL in self-relative form. |

1. Message 10: Success: Security descriptor of the account object has been updated.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarSetSecurityObject | Status | STATUS\_SUCCESS |

1. Message 11: Retrieve the Locally Unique Identifier (LUID) that the server assigns to the "SeTcbPrivilege" [**privilege**](#gt_d8092e10-b227-4b44-b015-511bb8178940).

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarLookupPrivilegeValue | PolicyHandle | [Implementation-specific value returned in Step 2.] |
| Send  LsarLookupPrivilegeValue | Name | "SeTcbPrivilege" |

1. Message 12: Success: Return the [**LUID**](#gt_96b64af9-1896-4bde-b988-54d469c5affd) of SeTcbPrivilege.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarLookupPrivilegeValue | Status | STATUS\_SUCCESS |
| Receive  LsarLookupPrivilegeValue | Value | The LUID assigned by the server to SeTcbPrivilege. |

1. Message 13: Add a privilege to the account object.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarAddPrivilegesToAccount | AccountHandle | [Implementation-specific value returned in Step 6.] |
| Send  LsarAddPrivilegesToAccount | Privileges | A LSAPR\_PRIVILEGE\_SET structure containing one privilege (the LUID of which was returned in Step 12). |

1. Message 14: Success: Privilege has been added to the account object.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarAddPrivilegesToAccount | Status | STATUS\_SUCCESS |

1. Message 15: Add a system access right to the account object.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarSetSystemAccessAccount | AccountHandle | [Implementation-specific value returned in Step 6.] |
| Send  LsarSetSystemAccessAccount | SystemAccess | An unsigned long value with the POLICY\_MODE\_NETWORK flag set |

1. Message 16: Success: Access right has been recorded.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarSetSystemAccessAccount | Status | STATUS\_SUCCESS |

1. Message 17: Done with this account object: Close the handle.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarClose | ObjectHandle | [Implementation-specific value returned in Step 6.] |

1. Message 18: Success: Account objects handle has been closed.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarClose | Status | STATUS\_SUCCESS |

1. Message 19: Done with the policy object: Close the handle.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarClose | ObjectHandle | [Implementation-specific value returned in Step 2.] |

1. Message 20: Success: Policy object has been closed.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarClose | Status | STATUS\_SUCCESS |

## Manipulating Secret Objects

This section illustrates a message exchange pertaining to [**secret objects**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d).

1. Message 1: Open the policy object.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarOpenPolicy2 | SystemName | "Arbitrary String" |
| Send  LsarOpenPolicy2 | ObjectAttributes | Ignored, except for the **RootDirectory** field, which is NULL. |
| Send  LsarOpenPolicy2 | DesiredAccess | POLICY\_VIEW\_LOCAL\_INFORMATION | POLICY\_CREATE\_SECRET |

1. Message 2: Success: Policy object opened successfully.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarOpenPolicy2 | Status | STATUS\_SUCCESS |
| Receive  LsarOpenPolicy2 | PolicyHandle | [Implementation-specific value] |

1. Message 3: Attempt to create a secret objects with name "NL$".

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarCreateSecret | PolicyHandle | [Implementation-specific value returned in Step 2.] |
| Send  LsarCreateSecret | Secretname | "NL$" |
| Send  LsarCreateSecret | DesiredAccess | SECRET\_SET\_VALUE |

1. Message 4: Failure: Secret name "NL$" is a reserved prefix name and cannot be used.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarCreateSecret | Status | STATUS\_INVALID\_PARAMETER |
| Receive  LsarCreateSecret | SecretHandle | NULL |

1. Message 5: Attempt to create a secret object with name "MyBigSecret".

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarCreateSecret | PolicyHandle | [Implementation-specific value returned in Step 2.] |
| Send  LsarCreateSecret | Secretname | "MyBigSecret" |
| Send  LsarCreateSecret | DesiredAccess | SECRET\_SET\_VALUE |

1. Message 6: Success: Secret created.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarCreateSecret | Status | STATUS\_SUCCESS |
| Receive  LsarCreateSecret | SecretHandle | [Implementation-specific value] |

1. Message 7: Set the value of the secret object.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarSetSecret | SecretHandle | [Implementation-specific value returned in Step 6.] |
| Send  LsarSetSecret | EncryptedCurrentValue | Byte BLOB value encrypted with session key. |
| Send  LsarSetSecret | EncryptedOldValue | NULL |

1. Message 8: Success: Secret value set.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarSetSecret | Status | STATUS\_SUCCESS |

1. Message 9: Done with this secret; close the handle.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarClose | ObjectHandle | [Implementation-specific value returned in Step 6.] |

1. Message 10: Success: Secret handle has been closed.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarClose | Status | STATUS\_SUCCESS |

1. Message 11: Done with the policy handle; close the handle.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarClose | ObjectHandle | [Implementation-specific value returned in Step 2.] |

1. Message 12: Success: Policy handle has been closed.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarClose | Status | STATUS\_SUCCESS |

## Manipulating Trusted Domain Objects

This section illustrates a message exchange pertaining to [**trusted domain objects**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4).

1. Message 1: Open the policy object.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarOpenPolicy2 | SystemName | "Arbitrary String" |
| Send  LsarOpenPolicy2 | ObjectAttributes | Ignored, except for the **RootDirectory** field, which is NULL. |
| Send  LsarOpenPolicy2 | DesiredAccess | POLICY\_VIEW\_LOCAL\_INFORMATION |

1. Message 2: Success; return the policy object handle.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarOpenPolicy2 | Status | STATUS\_SUCCESS |
| Receive  LsarOpenPolicy2 | PolicyHandle | [Implementation-specific value] |

1. Message 3: Enumerate trusted domain objects.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarEnumerateTrustedDomainsEx | PolicyHandle | [Implementation-specific value returned in Step 2.] |
| Send  LsarEnumerateTrustedDomainsEx | EnumerationContext | 0 |
| Send  LsarEnumerateTrustedDomainsEx | PreferredMaximumLength | 0x100 |

1. Message 4: Success; return some trusted domain objects, with more to come.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarEnumerateTrustedDomainsEx | Status | STATUS\_MORE\_ENTRIES |
| Receive  LsarEnumerateTrustedDomainsEx | EnumerationContext | [Implementation-specific value] |
| Receive  LsarEnumerateTrustedDomainsEx | TrustedDomainInformation | EntriesRead: 2  EnumerationBuffer: Contains information about two different trusted domain objects. |

1. Message 5: Finish enumerating the trusted domain objects.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarEnumerateTrustedDomainsEx | PolicyHandle | [Implementation-specific value returned in Step 2.] |
| Send  LsarEnumerateTrustedDomainsEx | EnumerationContext | [Value returned in Step 4.] |
| Send  LsarEnumerateTrustedDomainsEx | PreferredMaximumLength | 0x10000 |

1. Message 6: Success; all trusted domain objects have been enumerated.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarEnumerateTrustedDomainsEx | Status | STATUS\_NO\_MORE\_ENTRIES |
| Receive  LsarEnumerateTrustedDomainsEx | EnumerationContext | [Implementation-specific value] |
| Receive  LsarEnumerateTrustedDomainsEx | TrustedDomainInformation | EntriesRead: 3  EnumerationBuffer: Contains information about three different trusted domain objects. |

1. Message 7: Open a trusted domain object by name.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarOpenTrustedDomainByName | PolicyHandle | [Implementation-specific value returned in Step 2.] |
| Send  LsarOpenTrustedDomainByName | TrustedDomainName | [One of the [**DNS names**](#gt_102a36e2-f66f-49e2-bee3-558736b2ecd5) returned in Step 4 or Step 6.] |
| Send  LsarOpenTrustedDomainByName | DesiredAccess | POLICY\_TRUST\_ADMIN |

1. Message 8: Success; the trusted domain object has been opened successfully.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarOpenTrustedDomainByName | Status | STATUS\_SUCCESS |
| Receive  LsarOpenTrustedDomainByName | TrustedDomainHandle | [Implementation-specific value] |

1. Message 9: Done with this trusted domain object: Close the handle.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarClose | ObjectHandle | [Implementation-specific value returned in Step 8.] |

1. Message 10: Success: Trusted domain object has been closed.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarClose | Status | STATUS\_SUCCESS |

1. Message 11: Done with the policy object: Close the handle.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Send  LsarClose | ObjectHandle | [Implementation-specific value returned in Step 2.] |

1. Message 12: Success: Policy object has been closed.

| Direction and method | Parameter field | Parameter value |
| --- | --- | --- |
| Receive  LsarClose | Status | STATUS\_SUCCESS |

## Structure Example of LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB

The following is an annotated dump of [LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB](#Section_da8f32a10a164194810d06cc0698595a).

1. 00000000 93 1e 54 57 83 78 c6 c1 15 f1 13 85 3d 93 18 1d ..TW.x......=...
2. 00000010 b4 eb ee 6b fa 79 f5 2e 8d cb b4 e3 e3 54 8a 81 ...k.y.......T..
3. 00000020 b6 38 0f 6c 4d 6b 2b 36 4f a5 ce d8 82 44 52 23 .8.lMk+6O....DR#
4. 00000030 fc 40 b4 fd e2 71 78 95 f4 d5 29 e1 11 7a 8c 67 .@...qx...)..z.g
5. 00000040 2f 3d 69 a3 54 cd 47 79 ca 3f a8 a9 4f 08 85 9d /=i.T.Gy.?..O...
6. 00000050 93 fb 56 0b db 84 9e bb da c4 fd 58 3d 88 55 c7 ..V........X=.U.
7. 00000060 bb 5a 2d aa e3 26 23 a5 12 b1 1c 23 1a aa 72 26 .Z-..&#....#..r&
8. 00000070 9f 57 b2 89 be 37 ec 32 83 25 6c fe 7c ae 09 2b .W...7.2.%l.|..+
9. 00000080 27 15 20 01 c2 7a d3 2a e1 e5 5e 0c 16 17 10 4d '. ..z.\*..^....M
10. 00000090 6a dc 9b 3e 09 43 5a 66 8e 17 4c 27 d1 40 9b 19 j..>.CZf..L'.@..
11. 000000a0 82 ab d4 81 07 83 78 98 78 a1 f8 2a b2 9b 7c 5f ......x.x..\*..|\_
12. 000000b0 81 6f 11 37 e1 e5 90 4b 47 b4 0d 9a ac 3b 35 40 .o.7...KG....;5@
13. 000000c0 79 45 04 0e 59 07 67 ea f2 ea 57 c2 5d 25 03 94 yE..Y.g...W.]%..
14. 000000d0 46 35 7c 1f 01 e4 89 d9 1b 9d fe 94 e8 9e 3f 4e F5|...........?N
15. 000000e0 b1 18 43 2f 27 a7 f1 0a ff 1d 42 e2 ce 54 f9 2a ..C/'.....B..T.\*
16. 000000f0 b9 b2 43 81 f0 ed 22 2d e0 a0 37 2a c1 19 67 f0 ..C..."-..7\*..g.
17. 00000100 fa 5f 37 0a 9f 58 90 77 eb 0a 95 1c fe 5f a0 e7 .\_7..X.w.....\_..
18. 00000110 dc 4f 28 fa 18 d7 22 23 9b 54 e7 fd e5 ed 67 a2 .O(..."#.T....g.
19. 00000120 da a4 3d cb 0b f3 5a ce e9 dd de 0b d6 e7 e5 91 ..=...Z.........
20. 00000130 92 20 8f ac 2f bc be 11 55 b1 5e 0a 79 ed 00 4a . ../...U.^.y..J
21. 00000140 e6 94 34 8e 29 09 ef b5 2e 36 62 73 84 4d 4a 77 ..4.)....6bs.MJw
22. 00000150 3f df 9a 6f 4f 3c 3e 1a 11 e7 1c 8d 84 43 2e 1a ?..oO<>......C..
23. 00000160 aa 59 88 96 47 0b f0 6d 29 27 7b 68 c2 7b 2e be .Y..G..m)'{h.{..
24. 00000170 03 07 43 bf 8a 96 80 30 b9 1c 1e 36 e4 c0 d1 a6 ..C....0...6....
25. 00000180 a0 35 75 71 d5 ac f1 a3 1c d4 29 ee 40 50 68 93 .5uq......).@Ph.
26. 00000190 02 e7 aa 96 43 c2 46 1d cd f3 ba 9c 94 ff f6 74 ....C.F........t
27. 000001a0 3c 19 5f f6 1d 1c 11 42 ff 3e 34 0b 94 48 dc de <.\_....B.>4..H..
28. 000001b0 5d 74 7e 33 d8 cd 58 20 20 38 1a d5 e4 b6 fd 1d ]t~3..X 8......
29. 000001c0 c3 a7 ef 40 82 11 4c dc 2b 7e b4 ea 1a 85 ce f8 ...@..L.+~......
30. 000001d0 87 a3 ed 20 ff 50 4f ee 6c c3 b1 4c 2a 17 96 61 ... .PO.l..L\*..a
31. 000001e0 bc 5b 5e 17 52 d0 92 9e 11 70 d0 1d 98 a7 56 fa .[^.R....p....V.
32. 000001f0 69 95 6c 78 22 34 70 03 75 77 60 c5 3b 2f 42 e0 i.lx"4p.uw`.;/B.
33. 00000200 01 00 00 00 0c 00 00 00 30 00 00 00 49 42 b8 1c ........0...IB..
34. 00000210 b4 55 c9 01 02 00 00 00 11 00 00 00 4f 75 74 67 .U..........Outg
35. 00000220 6f 69 6e 67 41 75 74 68 49 6e 66 6f 00 00 00 00 oingAuthInfo....
36. 00000230 49 42 b8 1c b4 55 c9 01 02 00 00 00 19 00 00 00 IB...U..........
37. 00000240 4f 75 74 67 6f 69 6e 67 50 72 65 76 69 6f 75 73 OutgoingPrevious
38. 00000250 41 75 74 68 49 6e 66 6f 00 00 00 00 01 00 00 00 AuthInfo........
39. 00000260 0c 00 00 00 30 00 00 00 49 42 b8 1c b4 55 c9 01 ....0...IB...U..
40. 00000270 02 00 00 00 11 00 00 00 49 6e 63 6f 6d 69 6e 67 ........Incoming
41. 00000280 41 75 74 68 49 6e 66 6f 00 00 00 00 49 42 b8 1c AuthInfo....IB..
42. 00000290 b4 55 c9 01 02 00 00 00 19 00 00 00 49 6e 63 6f .U..........Inco
43. 000002a0 6d 69 6e 67 50 72 65 76 69 6f 75 73 41 75 74 68 mingPreviousAuth
44. 000002b0 49 6e 66 6f 00 00 00 00 5c 00 00 00 5c 00 00 00 Info....\...\...

The LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB leads with 512 bytes of random data:

1. 00000000 93 1e 54 57 83 78 c6 c1 15 f1 13 85 3d 93 18 1d ..TW.x......=...
2. 00000010 b4 eb ee 6b fa 79 f5 2e 8d cb b4 e3 e3 54 8a 81 ...k.y.......T..
3. 00000020 b6 38 0f 6c 4d 6b 2b 36 4f a5 ce d8 82 44 52 23 .8.lMk+6O....DR#
4. 00000030 fc 40 b4 fd e2 71 78 95 f4 d5 29 e1 11 7a 8c 67 .@...qx...)..z.g
5. 00000040 2f 3d 69 a3 54 cd 47 79 ca 3f a8 a9 4f 08 85 9d /=i.T.Gy.?..O...
6. 00000050 93 fb 56 0b db 84 9e bb da c4 fd 58 3d 88 55 c7 ..V........X=.U.
7. 00000060 bb 5a 2d aa e3 26 23 a5 12 b1 1c 23 1a aa 72 26 .Z-..&#....#..r&
8. 00000070 9f 57 b2 89 be 37 ec 32 83 25 6c fe 7c ae 09 2b .W...7.2.%l.|..+
9. 00000080 27 15 20 01 c2 7a d3 2a e1 e5 5e 0c 16 17 10 4d '. ..z.\*..^....M
10. 00000090 6a dc 9b 3e 09 43 5a 66 8e 17 4c 27 d1 40 9b 19 j..>.CZf..L'.@..
11. 000000a0 82 ab d4 81 07 83 78 98 78 a1 f8 2a b2 9b 7c 5f ......x.x..\*..|\_
12. 000000b0 81 6f 11 37 e1 e5 90 4b 47 b4 0d 9a ac 3b 35 40 .o.7...KG....;5@
13. 000000c0 79 45 04 0e 59 07 67 ea f2 ea 57 c2 5d 25 03 94 yE..Y.g...W.]%..
14. 000000d0 46 35 7c 1f 01 e4 89 d9 1b 9d fe 94 e8 9e 3f 4e F5|...........?N
15. 000000e0 b1 18 43 2f 27 a7 f1 0a ff 1d 42 e2 ce 54 f9 2a ..C/'.....B..T.\*
16. 000000f0 b9 b2 43 81 f0 ed 22 2d e0 a0 37 2a c1 19 67 f0 ..C..."-..7\*..g.
17. 00000100 fa 5f 37 0a 9f 58 90 77 eb 0a 95 1c fe 5f a0 e7 .\_7..X.w.....\_..
18. 00000110 dc 4f 28 fa 18 d7 22 23 9b 54 e7 fd e5 ed 67 a2 .O(..."#.T....g.
19. 00000120 da a4 3d cb 0b f3 5a ce e9 dd de 0b d6 e7 e5 91 ..=...Z.........
20. 00000130 92 20 8f ac 2f bc be 11 55 b1 5e 0a 79 ed 00 4a . ../...U.^.y..J
21. 00000140 e6 94 34 8e 29 09 ef b5 2e 36 62 73 84 4d 4a 77 ..4.)....6bs.MJw
22. 00000150 3f df 9a 6f 4f 3c 3e 1a 11 e7 1c 8d 84 43 2e 1a ?..oO<>......C..
23. 00000160 aa 59 88 96 47 0b f0 6d 29 27 7b 68 c2 7b 2e be .Y..G..m)'{h.{..
24. 00000170 03 07 43 bf 8a 96 80 30 b9 1c 1e 36 e4 c0 d1 a6 ..C....0...6....
25. 00000180 a0 35 75 71 d5 ac f1 a3 1c d4 29 ee 40 50 68 93 .5uq......).@Ph.
26. 00000190 02 e7 aa 96 43 c2 46 1d cd f3 ba 9c 94 ff f6 74 ....C.F........t
27. 000001a0 3c 19 5f f6 1d 1c 11 42 ff 3e 34 0b 94 48 dc de <.\_....B.>4..H..
28. 000001b0 5d 74 7e 33 d8 cd 58 20 20 38 1a d5 e4 b6 fd 1d ]t~3..X 8......
29. 000001c0 c3 a7 ef 40 82 11 4c dc 2b 7e b4 ea 1a 85 ce f8 ...@..L.+~......
30. 000001d0 87 a3 ed 20 ff 50 4f ee 6c c3 b1 4c 2a 17 96 61 ... .PO.l..L\*..a
31. 000001e0 bc 5b 5e 17 52 d0 92 9e 11 70 d0 1d 98 a7 56 fa .[^.R....p....V.
32. 000001f0 69 95 6c 78 22 34 70 03 75 77 60 c5 3b 2f 42 e0 i.lx"4p.uw`.;/B.

The data following that is for **CountOutgoingAuthInfos**, in little-endian byte order:

1. 00000200 01 00 00 00 ....

This indicates that there is one entry present in the **CurrentOutgoingAuthInfos** field.

The data following that is for **ByteOffsetCurrentOutgoingAuthInfo**, in little-endian byte order:

1. 00000204 0c 00 00 00 ....

This means that the byte offset from the beginning of **CountOutgoingAuthInfos** to the start of the **CurrentOutgoingAuthInfos** field is 0x0000000c.

The data following that is for **ByteOffsetPreviousOutgoingAuthInfo**, in little-endian byte order:

1. 00000208 30 00 00 00 0...

This means that the byte offset from the beginning of **CountOutgoingAuthInfos** to the start of the **PreviousOutgoingAuthInfos** field is 0x00000030, so the size of **CurrentOutgoingAuthInfos** is 0x30 – 0xc = 0x24 bytes.

The data following that is for **CurrentOutgoingAuthInfos**; the last 3 padding bytes are for data alignment purposes:

1. 0000020c 49 42 b8 1c IB..
2. 00000210 b4 55 c9 01 02 00 00 00 11 00 00 00 4f 75 74 67 .U..........Outg
3. 00000220 6f 69 6e 67 41 75 74 68 49 6e 66 6f 00 00 00 00 oingAuthInfo....

This is an array of **CountOutgoingAuthInfos** of [LSAPR\_AUTH\_INFORMATION (section 2.2.7.17)](#Section_cedb0d1bc7c0448099fc279b06f22a0c) entries in self-relative format.

The data following that is for **PreviousOutgoingAuthInfos**; the last 3 padding bytes are for data alignment purposes:

1. 00000230 49 42 b8 1c b4 55 c9 01 02 00 00 00 19 00 00 00 IB...U..........
2. 00000240 4f 75 74 67 6f 69 6e 67 50 72 65 76 69 6f 75 73 OutgoingPrevious
3. 00000250 41 75 74 68 49 6e 66 6f 00 00 00 00 AuthInfo....

This is an array of **CountOutgoingAuthInfos** of LSAPR\_AUTH\_INFORMATION entries in self-relative format.

The data following that is for **CountIncomingAuthInfos**, in little-endian byte order:

1. 0000025c 01 00 00 00 ....

This means there is one entry present in the **CountIncomingAuthInfos** field.

The data following that is for **ByteOffsetCurrentIncomingAuthInfo**, in little-endian byte order:

1. 00000260 0c 00 00 00 ....

This means that the byte offset from the beginning of **CountIncomingAuthInfos** to the start of the **CurrentIncomingAuthInfos** field is 0x0000000c.

The data following that is for **ByteOffsetPreviousIncomingAuthInfo**, in little-endian byte order:

1. 00000264 30 00 00 00 0...

This means that the byte offset from the beginning of **CountIncomingAuthInfos** to the start of the **PreviousIncomingAuthInfos** field is 0x00000030, so the size of **CurrentIncomingAuthInfos** is 0x30 – 0xc = 0x24 bytes.

The data following that is for **CurrentIncomingAuthInfos**; the last 3 padding bytes are for data alignment purposes:

1. 00000268 49 42 b8 1c b4 55 c9 01 IB...U..
2. 00000270 02 00 00 00 11 00 00 00 49 6e 63 6f 6d 69 6e 67 ........Incoming
3. 00000280 41 75 74 68 49 6e 66 6f 00 00 00 00 AuthInfo....

This is an array of **CountIncomingAuthInfos** of LSAPR\_AUTH\_INFORMATION entries in self-relative format.

The data following that is for **PreviousIncomingAuthInfos**; the last 3 padding bytes are for data alignment purposes:

1. 0000028c 49 42 b8 1c IB..
2. 00000290 b4 55 c9 01 02 00 00 00 19 00 00 00 49 6e 63 6f .U..........Inco
3. 000002a0 6d 69 6e 67 50 72 65 76 69 6f 75 73 41 75 74 68 mingPreviousAuth
4. 000002b0 49 6e 66 6f 00 00 00 00 Info....

This is an array of **CountIncomingAuthInfos** of LSAPR\_AUTH\_INFORMATION entries in self-relative format.

The data following that is for **OutgoingAuthInfoSize**, in little-endian byte order:

1. 000002b8 5c 00 00 00 \...

This means that the size, in bytes, of the sub-portion of the structure from the beginning of the **CountOutgoingAuthInfos** field through the end of the of the **PreviousOutgoingAuthInfos** field is 0x0000005c.

The data following that is for **IncomingAuthInfoSize**, in little-endian byte order:

1. 000002bc 5c 00 00 00 \...

This means that the size, in bytes, of the sub-portion of the structure from the beginning of the **CountIncomingAuthInfos** field through the end of the of the **PreviousIncomingAuthInfos** field is 0x0000005c.

# Security

## Security Considerations for Implementers

Usage of [**RC4**](#gt_d57eac33-f561-4a08-b148-dfcf29cfb4d8) is specified in section [5.1.1](#Section_1f5bd3edcfdd42aba2acf0786082bb21). This protocol employs an implementation that reuses RC4 key stream, which subjects it to Xor and other cryptanalysis attacks. This vulnerability is applicable when multiple RC4-encrypted [**opnum**](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7) requests are made over the same transport session, as specified in section [2.1](#Section_64ea7ac432ef44f6ab51ea2b5a1c2390).

Usage of Data Encryption Standard (DES) in Electronic Code Book (ECB) mode is specified in section [5.1.2](#Section_cce8ae7801384129954ec65e0c0bffed). This algorithm is considered inadequate for maintaining confidentiality considering the efficiency of brute-force and cryptanalysis attacks that are enabled by using year 2006, off-the-shelf computer hardware.

The session key for sections 5.1.1 and 5.1.2 is obtained from the [**SMB**](#gt_09dbec39-5e75-4d9a-babf-1c9f1d499625) transport, as specified in section 2.1. The session key is obtained from the SMB transport every time a message that needs encryption is to be sent or a message that needs decryption is to be received.

### RC4 Cipher Usage

Implementations of this protocol protect the [LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB](#Section_da8f32a10a164194810d06cc0698595a) structure by encrypting the data referenced by that structure's **AuthBlob** field. The [**RC4**](#gt_d57eac33-f561-4a08-b148-dfcf29cfb4d8) algorithm is used to encrypt the data on request (and reply) and decrypt the data on receipt. The key, required during runtime by the RC4 algorithm, is the 16-byte key specified by the method that uses this structure (for example, see section [3.1.4.7.10](#Section_cc86a55db61948fd998a65cca15efeb9)). The size of data (the **AuthSize** field of LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB) must remain unencrypted.

### Secret Encryption and Decryption

This cipher is used to provide confidentiality of wire traffic for operations that reference this section.

The encrypt\_secret routine is used to encrypt a cleartext value into ciphertext prior to transmission. The decrypt\_secret routine is used to decrypt a ciphertext value into cleartext after receipt. The appropriate mode is selected based on the requirements of the interface.

The definitions of **des\_ecb\_lm\_dec** and **des\_ecb\_lm\_enc** are specified in section [5.1.3](#Section_32a2c8afdc6e4662918def333d570dd2).

1. encrypt\_secret(input : LSA\_UNICODE\_STRING, sessionkey : byte[16],
2. output : LSA\_UNICODE\_STRING)
3. {
4. LET blocklen be 8
5. LET keyindex be 0
6. // Set version, length
7. // temporary buffer.
8. LET buffer be an array of blocklen bytes
9. DECLEARE Version as ULONG
10. SET Version to 1
11. SET buffer to input->length
12. SET (buffer + 4) to Version
13. CALL des\_ecb\_lm\_enc(buffer, sessionkey[keyindex],
14. output->buffer)
15. INCREMENT output->buffer by blocklen
16. INCREMENT output->length by blocklen
17. SET keyindex to AdvanceKey(keyindex)
18. LET remaining be input->length
19. WHILE remaining > blocklen
20. CALL des\_ecb\_lm\_enc(input->buffer,
21. sessionkey[keyindex], output->buffer)
22. INCREMENT input->buffer by blocklen
23. INCREMENT output->buffer by blocklen
24. INCREMENT output->length by blocklen
25. SET keyindex to AdvanceKey(keyindex)
26. DECREMENT remaining by blocklen
27. ENDWHILE
28. IF (remaining > 0) THEN
29. // zero pad the last block.
30. SET bytes in buffer to 0
31. COPY remaining bytes from input->buffer to buffer
33. CALL des\_ecb\_lm\_enc(buffer, sessionkey[keyindex],
34. output->buffer)
35. INCREMENT output->length by blocklen
36. ENDIF
37. }
38. decrypt\_secret(input : LSA\_UNICODE\_STRING, sessionkey : byte[16],
39. output : LSA\_UNICODE\_STRING)
40. {
41. LET keyindex be 0
42. LET blocklen be 8
43. // Check version, get clear length.
44. CALL des\_ecb\_lm\_dec(input->buffer, sessionkey[keyindex],
45. output->buffer)
46. LET outputlength be output[0]
47. LET version be output[1]
48. IF (version ≠ 1) THEN // version check
49. FAIL
50. ENDIF
52. INCREMENT input->buffer by blocklen
53. SET keyindex to AdvanceKey(keyindex)
54. LET remaining be outputlength
55. WHILE remaining > blocklen
56. CALL des\_ecb\_lm\_dec(input->buffer,
57. sessionkey[keyindex], output->buffer)
59. INCREMENT input->buffer by blocklen
60. INCREMENT output->buffer by blocklen
61. SET keyindex to AdvanceKey(keyindex)
62. DECREMENT remaining by blocklen
63. ENDWHILE
64. IF (remaining > 0) THEN
65. CALL des\_ecb\_lm\_dec(input->buffer,
66. sessionkey[keyindex], output->buffer)
67. ENDIF
68. SET output->length to outputlength
69. }
70. ULONG AdvanceKey(pos : ULONG)
71. {
72. LET KeyBlockLen be 7;
73. DECLARE overrun as ULONG
74. DECLARE currpos as ULONG;
75. DECLARE nextpos as ULONG;
76. LET pos = pos + KeyBlockLen;
77. LET currpos = pos;
78. LET nextpos = pos + KeyBlockLen;
79. IF (nextpos > sizeof(sessionKey)) THEN
80. LET overrun = nextpos- sizeof(sessionKey);
81. LET currpos = (KeyBlockLen - overrun);
82. ENDIF
83. RETURN currpos;
84. }

### DES-ECB-LM Cipher Definition

des\_ecb\_lm\_dec utilizes DES-ECB-LM in cipher-mode decryption.

des\_ecb\_lm\_enc utilizes DES-ECB-LM in cipher-mode encryption.

DES-ECB-LM is defined as follows.

1. des\_ecb\_lm( input:byte[8], encryptionKey: byte[8],
2. output:byte[8])
3. InputKey:byte[7]
4. OutputKey:byte[8]
5. Let InputKey be the first 7 bytes of encryptionKey [0-6]
6. OutputKey[0] = InputKey[0] >> 0x01;
7. OutputKey[1] = ((InputKey[0]&0x01)<<6) | (InputKey[1]>>2);
8. OutputKey[2] = ((InputKey[1]&0x03)<<5) | (InputKey[2]>>3);
9. OutputKey[3] = ((InputKey[2]&0x07)<<4) | (InputKey[3]>>4);
10. OutputKey[4] = ((InputKey[3]&0x0F)<<3) | (InputKey[4]>>5);
11. OutputKey[5] = ((InputKey[4]&0x1F)<<2) | (InputKey[5]>>6);
12. OutputKey[6] = ((InputKey[5]&0x3F)<<1) | (InputKey[6]>>7);
13. OutputKey[7] = InputKey[6] & 0x7F;
14. ((unsigned long\*)OutputKey)[0] <<= 1;
15. ((unsigned long\*)OutputKey)[1] <<= 1;
16. ((unsigned long\*)OutputKey)[0] &= 0xfefefefe;
17. ((unsigned long\*)OutputKey)[1] &= 0xfefefefe;
18. Let the left-most bit of OutputKey be the parity bit. That is,
19. if the sum of the other 7 bits is odd, the parity bit is zero;
20. otherwise the parity bit is one. The processing starts at the
21. left-most bit of OutputKey.
22. des\_ecb( input, OutputKey, output )
23. END

The algorithm des\_ecb is the Data Encryption Standard (DES) encryption in Electronic Code Book (ECB) mode, as specified in [[FIPS81]](https://go.microsoft.com/fwlink/?LinkId=89874).

### Encryption and Decryption Examples

This section provides an encryption and decryption example of the algorithms specified in section [5.1.2](#Section_CCE8AE7801384129954EC65E0C0BFFED).

#### Encryption Example

This section provides an example of how the encrypt\_secret routine defined in section [5.1.2](#Section_CCE8AE7801384129954EC65E0C0BFFED) encrypts a given cleartext value into a ciphertext using a session key, and given the following parameters:

| Parameters | Value |
| --- | --- |
| *input* | 50 00 61 00 73 00 73 00 77 00 6f 00 72 00 64 00 31 00 32 00 33 00 |
| *sessionkey* | 4e 98 c9 10 b2 a9 88 d7 92 fb 5a a3 3e 8e f7 86 |

The following table describes the values *keyindex* (defined in **encrypt\_secret**) and *input*, *InputKey*, *OutputKey*, and *output* (defined in **des\_ecb\_lm**) after successive calls to **des\_ecb\_lm\_enc**.

| keyindex | input | InputKey | OutputKey | output |
| --- | --- | --- | --- | --- |
| 0 | 16 00 00 00 01 00 00 00 | 4e 98 c9 10 b2 a9 88 | 4f 4c 32 23 0b 94 a7 10 | af 78 44 03  fb a0 92 27 |
| 7 | 50 00 61 00 73 00 73 00 | d7 92 fb 5a a3 3e 8e | d6 c8 bf 6b ab 19 fb 1c | 8c 95 fc 7e  88 56 4c cd |
| 2 | 77 00 6f 00 72 00 64 00 | c9 10 b2 a9 88 d7 92 | c8 89 2c 54 98 46 5e 25 | 7c d0 c2 41  da 6f 14 41 |
| 9 | 31 00 32 00 33 00 00 00 | fb 5a a3 3e 8e f7 86 | fb ad a8 67 e9 76 df 0d | 56 26 cd a5  81 e9 22 3d |

The *output* variable fields are:

| Parameters | Value |
| --- | --- |
| *output->buffer* | af 78 44 03 fb a0 92 27 8c 95 fc 7e 88 56 4c cd 7c d0 c2 41 da 6f 14 41 56 26 cd a5 81 e9 22 3d |
| *output->length* | 0x20 |

#### Decryption Example

This section provides an example of how the **decrypt\_secret** routine defined in section [5.1.2](#Section_CCE8AE7801384129954EC65E0C0BFFED) decrypts a given ciphertext value into cleartext using a session key, and given the following parameters:

| Parameters | Value |
| --- | --- |
| *Input* | af 78 44 03 fb a0 92 27 8c 95 fc 7e 88 56 4c cd 7c d0 c2 41 da 6f 14 41 56 26 cd a5 81 e9 22 3d |
| *sessionkey* | 4e 98 c9 10 b2 a9 88 d7 92 fb 5a a3 3e 8e f7 86 |

The following table describes the values *keyindex* (defined in **decrypt\_secret**) and *input*, *InputKey*, *OutputKey*, and *output* (defined in **des\_ecb\_lm**) after successive calls to **des\_ecb\_lm\_dec**.

| keyindex | input | InputKey | OutputKey | output |
| --- | --- | --- | --- | --- |
| 0 | af 78 44 03 fb a0 92 27 | 4e 98 c9 10 b2 a9 88 | 4f 4c 32 23 0b 94 a7 10 | 16 00 00 00  01 00 00 00 |
| 7 | 8c 95 fc 7e 88 56 4c cd | d7 92 fb 5a a3 3e 8e | d6 c8 bf 6b ab 19 fb 1c | 50 00 61 00  73 00 73 00 |
| 2 | 7c d0 c2 41 da 6f 14 41 | c9 10 b2 a9 88 d7 92 | c8 89 2c 54 98 46 5e 25 | 77 00 6f 00  72 00 64 00 |
| 9 | 56 26 cd a5 81 e9 22 3d | fb 5a a3 3e 8e f7 86 | fb ad a8 67 e9 76 df 0d | 31 00 32 00  33 00 00 00 |

The *output* variable fields are:

| Parameters | Value |
| --- | --- |
| *output->buffer* | 50 00 61 00 73 00 73 00 77 00 6f 00 72 00 64 00 31 00 32 00 33 00 00 00 |
| *output->length* | 0x16 |

## Index of Security Parameters

| Security parameter | Section |
| --- | --- |
| Usage of [**RC4**](#gt_d57eac33-f561-4a08-b148-dfcf29cfb4d8) stream cipher | [5.1.1](#Section_1f5bd3edcfdd42aba2acf0786082bb21) |
| Usage of DES\_ECB\_LM | [5.1.2](#Section_cce8ae7801384129954ec65e0c0bffed) |

# Appendix A: Full IDL

For ease of implementation, the full IDL is provided, where ms-dtyp.idl is the IDL specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) Appendix A.

**Note**  The lsarpc interface is shared between this protocol and the Local Security Authority (Translation Methods) Remote Protocol [[MS-LSAT]](%5bMS-LSAT%5d.pdf#Section_1ba21e6fd8a9462c91534375f2020894). For convenience, the IDL definitions that appear below and the IDL definitions in [MS-LSAT] section 6 have been merged and are available for download. For more information, see [[MSFT-LSA-IDL]](https://go.microsoft.com/fwlink/?LinkId=150268).

1. import "ms-dtyp.idl";
2. [
3. uuid(12345778-1234-ABCD-EF00-0123456789AB),
4. version(0.0),
5. ms\_union,
6. pointer\_default(unique)
7. ]
8. interface lsarpc
9. {
10. //
11. // Type definitions.
12. //
13. //
14. // Start of common types.
15. //
16. typedef [context\_handle] void \* LSAPR\_HANDLE;
17. typedef unsigned char SECURITY\_CONTEXT\_TRACKING\_MODE,
18. \*PSECURITY\_CONTEXT\_TRACKING\_MODE;
19. typedef unsigned short SECURITY\_DESCRIPTOR\_CONTROL,
20. \*PSECURITY\_DESCRIPTOR\_CONTROL;
21. typedef struct \_STRING {
22. unsigned short Length;
23. unsigned short MaximumLength;
24. [size\_is(MaximumLength), length\_is(Length)]
25. char \* Buffer;
26. } STRING, \*PSTRING;
27. typedef struct \_LSAPR\_ACL {
28. unsigned char AclRevision;
29. unsigned char Sbz1;
30. unsigned short AclSize;
31. [size\_is(AclSize - 4)] unsigned char Dummy1[\*];
32. } LSAPR\_ACL, \*PLSAPR\_ACL;
33. typedef struct \_LSAPR\_SECURITY\_DESCRIPTOR {
34. unsigned char Revision;
35. unsigned char Sbz1;
36. SECURITY\_DESCRIPTOR\_CONTROL Control;
37. PRPC\_SID Owner;
38. PRPC\_SID Group;
39. PLSAPR\_ACL Sacl;
40. PLSAPR\_ACL Dacl;
41. } LSAPR\_SECURITY\_DESCRIPTOR, \*PLSAPR\_SECURITY\_DESCRIPTOR;
42. typedef enum \_SECURITY\_IMPERSONATION\_LEVEL {
43. SecurityAnonymous = 0,
44. SecurityIdentification = 1,
45. SecurityImpersonation = 2,
46. SecurityDelegation = 3
47. } SECURITY\_IMPERSONATION\_LEVEL, \* PSECURITY\_IMPERSONATION\_LEVEL;
48. typedef struct \_SECURITY\_QUALITY\_OF\_SERVICE {
49. unsigned long Length;
50. SECURITY\_IMPERSONATION\_LEVEL ImpersonationLevel;
51. SECURITY\_CONTEXT\_TRACKING\_MODE ContextTrackingMode;
52. unsigned char EffectiveOnly;
53. } SECURITY\_QUALITY\_OF\_SERVICE, \* PSECURITY\_QUALITY\_OF\_SERVICE;
54. typedef struct \_LSAPR\_OBJECT\_ATTRIBUTES {
55. unsigned long Length;
56. unsigned char \* RootDirectory;
57. PSTRING ObjectName;
58. unsigned long Attributes;
59. PLSAPR\_SECURITY\_DESCRIPTOR SecurityDescriptor;
60. PSECURITY\_QUALITY\_OF\_SERVICE SecurityQualityOfService;
61. } LSAPR\_OBJECT\_ATTRIBUTES, \*PLSAPR\_OBJECT\_ATTRIBUTES;
62. typedef struct \_LSAPR\_TRUST\_INFORMATION {
63. RPC\_UNICODE\_STRING Name;
64. PRPC\_SID Sid;
65. } LSAPR\_TRUST\_INFORMATION, \*PLSAPR\_TRUST\_INFORMATION;
66. //
67. // End of common types.
68. //
69. typedef enum \_POLICY\_INFORMATION\_CLASS {
70. PolicyAuditLogInformation = 1,
71. PolicyAuditEventsInformation,
72. PolicyPrimaryDomainInformation,
73. PolicyPdAccountInformation,
74. PolicyAccountDomainInformation,
75. PolicyLsaServerRoleInformation,
76. PolicyReplicaSourceInformation,
77. PolicyInformationNotUsedOnWire,
78. PolicyModificationInformation,
79. PolicyAuditFullSetInformation,
80. PolicyAuditFullQueryInformation,
81. PolicyDnsDomainInformation,
82. PolicyDnsDomainInformationInt,
83. PolicyLocalAccountDomainInformation,
84. PolicyLastEntry
85. } POLICY\_INFORMATION\_CLASS, \*PPOLICY\_INFORMATION\_CLASS;
86. typedef enum \_POLICY\_AUDIT\_EVENT\_TYPE {
87. AuditCategorySystem = 0,
88. AuditCategoryLogon,
89. AuditCategoryObjectAccess,
90. AuditCategoryPrivilegeUse,
91. AuditCategoryDetailedTracking,
92. AuditCategoryPolicyChange,
93. AuditCategoryAccountManagement,
94. AuditCategoryDirectoryServiceAccess,
95. AuditCategoryAccountLogon
96. } POLICY\_AUDIT\_EVENT\_TYPE, \*PPOLICY\_AUDIT\_EVENT\_TYPE;
97. typedef RPC\_UNICODE\_STRING LSA\_UNICODE\_STRING,
98. \*PLSA\_UNICODE\_STRING;
99. typedef struct \_POLICY\_AUDIT\_LOG\_INFO {
100. unsigned long AuditLogPercentFull;
101. unsigned long MaximumLogSize;
102. LARGE\_INTEGER AuditRetentionPeriod;
103. unsigned char AuditLogFullShutdownInProgress;
104. LARGE\_INTEGER TimeToShutdown;
105. unsigned long NextAuditRecordId;
106. } POLICY\_AUDIT\_LOG\_INFO, \*PPOLICY\_AUDIT\_LOG\_INFO;
107. typedef enum \_POLICY\_LSA\_SERVER\_ROLE {
108. PolicyServerRoleBackup = 2,
109. PolicyServerRolePrimary
110. } POLICY\_LSA\_SERVER\_ROLE, \*PPOLICY\_LSA\_SERVER\_ROLE;
111. typedef struct \_POLICY\_LSA\_SERVER\_ROLE\_INFO {
112. POLICY\_LSA\_SERVER\_ROLE LsaServerRole;
113. } POLICY\_LSA\_SERVER\_ROLE\_INFO, \*PPOLICY\_LSA\_SERVER\_ROLE\_INFO;
114. typedef struct \_POLICY\_MODIFICATION\_INFO {
115. LARGE\_INTEGER ModifiedId;
116. LARGE\_INTEGER DatabaseCreationTime;
117. } POLICY\_MODIFICATION\_INFO, \*PPOLICY\_MODIFICATION\_INFO;
118. typedef struct \_POLICY\_AUDIT\_FULL\_SET\_INFO {
119. unsigned char ShutDownOnFull;
120. } POLICY\_AUDIT\_FULL\_SET\_INFO,
121. \*PPOLICY\_AUDIT\_FULL\_SET\_INFO;
122. typedef struct \_POLICY\_AUDIT\_FULL\_QUERY\_INFO {
123. unsigned char ShutDownOnFull;
124. unsigned char LogIsFull;
125. } POLICY\_AUDIT\_FULL\_QUERY\_INFO,
126. \*PPOLICY\_AUDIT\_FULL\_QUERY\_INFO;
127. typedef enum \_POLICY\_DOMAIN\_INFORMATION\_CLASS {
128. PolicyDomainQualityOfServiceInformation = 1,
129. PolicyDomainEfsInformation = 2,
130. PolicyDomainKerberosTicketInformation = 3
131. } POLICY\_DOMAIN\_INFORMATION\_CLASS,
132. \*PPOLICY\_DOMAIN\_INFORMATION\_CLASS;
133. typedef struct \_POLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO {
134. unsigned long AuthenticationOptions;
135. LARGE\_INTEGER MaxServiceTicketAge;
136. LARGE\_INTEGER MaxTicketAge;
137. LARGE\_INTEGER MaxRenewAge;
138. LARGE\_INTEGER MaxClockSkew;
139. LARGE\_INTEGER Reserved;
140. } POLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO,
141. \*PPOLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO;
142. typedef struct \_TRUSTED\_POSIX\_OFFSET\_INFO {
143. unsigned long Offset;
144. } TRUSTED\_POSIX\_OFFSET\_INFO,
145. \*PTRUSTED\_POSIX\_OFFSET\_INFO;
146. typedef enum \_TRUSTED\_INFORMATION\_CLASS {
147. TrustedDomainNameInformation = 1,
148. TrustedControllersInformation,
149. TrustedPosixOffsetInformation,
150. TrustedPasswordInformation,
151. TrustedDomainInformationBasic,
152. TrustedDomainInformationEx,
153. TrustedDomainAuthInformation,
154. TrustedDomainFullInformation,
155. TrustedDomainAuthInformationInternal,
156. TrustedDomainFullInformationInternal,
157. TrustedDomainInformationEx2Internal,
158. TrustedDomainFullInformation2Internal,
159. TrustedDomainSupportedEncryptionTypes
160. } TRUSTED\_INFORMATION\_CLASS,
161. \*PTRUSTED\_INFORMATION\_CLASS;
162. typedef enum \_LSA\_FOREST\_TRUST\_RECORD\_TYPE {
163. ForestTrustTopLevelName = 0,
164. ForestTrustTopLevelNameEx = 1,
165. ForestTrustDomainInfo = 2
166. } LSA\_FOREST\_TRUST\_RECORD\_TYPE;
167. typedef struct \_LSA\_FOREST\_TRUST\_BINARY\_DATA {
168. [range(0, 131072)] unsigned long Length;
169. [size\_is( Length )] unsigned char \* Buffer;
170. } LSA\_FOREST\_TRUST\_BINARY\_DATA,
171. \*PLSA\_FOREST\_TRUST\_BINARY\_DATA;
172. typedef struct \_LSA\_FOREST\_TRUST\_DOMAIN\_INFO {
173. PRPC\_SID Sid;
174. LSA\_UNICODE\_STRING DnsName;
175. LSA\_UNICODE\_STRING NetbiosName;
176. } LSA\_FOREST\_TRUST\_DOMAIN\_INFO,
177. \*PLSA\_FOREST\_TRUST\_DOMAIN\_INFO;
178. typedef struct \_LSA\_FOREST\_TRUST\_RECORD {
179. unsigned long Flags;
180. LSA\_FOREST\_TRUST\_RECORD\_TYPE ForestTrustType;
181. LARGE\_INTEGER Time;
182. [switch\_type( LSA\_FOREST\_TRUST\_RECORD\_TYPE ),
183. switch\_is( ForestTrustType )]
184. union
185. {
186. [case( ForestTrustTopLevelName,
187. ForestTrustTopLevelNameEx )]
188. LSA\_UNICODE\_STRING TopLevelName;
189. [case( ForestTrustDomainInfo )]
190. LSA\_FOREST\_TRUST\_DOMAIN\_INFO DomainInfo;
191. [default] LSA\_FOREST\_TRUST\_BINARY\_DATA Data;
192. } ForestTrustData;
193. } LSA\_FOREST\_TRUST\_RECORD, \*PLSA\_FOREST\_TRUST\_RECORD;
194. typedef struct \_LSA\_FOREST\_TRUST\_INFORMATION {
195. [range(0,4000)] unsigned long RecordCount;
196. [size\_is( RecordCount )] PLSA\_FOREST\_TRUST\_RECORD \* Entries;
197. } LSA\_FOREST\_TRUST\_INFORMATION, \*PLSA\_FOREST\_TRUST\_INFORMATION;
198. typedef enum \_LSA\_FOREST\_TRUST\_COLLISION\_RECORD\_TYPE {
199. CollisionTdo = 0,
200. CollisionXref,
201. CollisionOther
202. } LSA\_FOREST\_TRUST\_COLLISION\_RECORD\_TYPE;
203. typedef struct \_LSA\_FOREST\_TRUST\_COLLISION\_RECORD {
204. unsigned long Index;
205. LSA\_FOREST\_TRUST\_COLLISION\_RECORD\_TYPE Type;
206. unsigned long Flags;
207. LSA\_UNICODE\_STRING Name;
208. } LSA\_FOREST\_TRUST\_COLLISION\_RECORD,
209. \*PLSA\_FOREST\_TRUST\_COLLISION\_RECORD;
210. typedef struct \_LSA\_FOREST\_TRUST\_COLLISION\_INFORMATION {
211. unsigned long RecordCount;
212. [size\_is( RecordCount )]
213. PLSA\_FOREST\_TRUST\_COLLISION\_RECORD \* Entries;
214. } LSA\_FOREST\_TRUST\_COLLISION\_INFORMATION,
215. \*PLSA\_FOREST\_TRUST\_COLLISION\_INFORMATION;
216. typedef LSAPR\_HANDLE \*PLSAPR\_HANDLE;
217. typedef struct \_LSAPR\_ACCOUNT\_INFORMATION {
218. PRPC\_SID Sid;
219. } LSAPR\_ACCOUNT\_INFORMATION, \*PLSAPR\_ACCOUNT\_INFORMATION;
220. typedef struct \_LSAPR\_ACCOUNT\_ENUM\_BUFFER {
221. unsigned long EntriesRead;
222. [size\_is(EntriesRead)] PLSAPR\_ACCOUNT\_INFORMATION Information;
223. } LSAPR\_ACCOUNT\_ENUM\_BUFFER, \*PLSAPR\_ACCOUNT\_ENUM\_BUFFER;
224. typedef struct \_LSAPR\_SR\_SECURITY\_DESCRIPTOR {
225. [range(0,262144)] unsigned long Length;
226. [size\_is(Length)] unsigned char \* SecurityDescriptor;
227. } LSAPR\_SR\_SECURITY\_DESCRIPTOR, \*PLSAPR\_SR\_SECURITY\_DESCRIPTOR;
228. typedef struct \_LSAPR\_LUID\_AND\_ATTRIBUTES {
229. LUID Luid;
230. unsigned long Attributes;
231. } LSAPR\_LUID\_AND\_ATTRIBUTES, \* PLSAPR\_LUID\_AND\_ATTRIBUTES;
232. typedef struct \_LSAPR\_PRIVILEGE\_SET {
233. [range(0,1000)] unsigned long PrivilegeCount;
234. unsigned long Control;
235. [size\_is(PrivilegeCount)] LSAPR\_LUID\_AND\_ATTRIBUTES Privilege[\*];
236. } LSAPR\_PRIVILEGE\_SET, \*PLSAPR\_PRIVILEGE\_SET;
237. typedef struct \_LSAPR\_POLICY\_PRIVILEGE\_DEF {
238. RPC\_UNICODE\_STRING Name;
239. LUID LocalValue;
240. } LSAPR\_POLICY\_PRIVILEGE\_DEF, \*PLSAPR\_POLICY\_PRIVILEGE\_DEF;
241. typedef struct \_LSAPR\_PRIVILEGE\_ENUM\_BUFFER {
242. unsigned long Entries;
243. [size\_is(Entries)] PLSAPR\_POLICY\_PRIVILEGE\_DEF Privileges;
244. } LSAPR\_PRIVILEGE\_ENUM\_BUFFER, \*PLSAPR\_PRIVILEGE\_ENUM\_BUFFER;
245. typedef struct \_LSAPR\_CR\_CIPHER\_VALUE {
246. [range(0, 131088)] unsigned long Length;
247. [range(0, 131088)] unsigned long MaximumLength;
248. [size\_is(MaximumLength), length\_is(Length)]
249. unsigned char \*Buffer;
250. } LSAPR\_CR\_CIPHER\_VALUE, \*PLSAPR\_CR\_CIPHER\_VALUE;
251. typedef struct \_LSAPR\_TRUSTED\_ENUM\_BUFFER {
252. unsigned long EntriesRead;
253. [size\_is(EntriesRead)] PLSAPR\_TRUST\_INFORMATION Information;
254. } LSAPR\_TRUSTED\_ENUM\_BUFFER, \*PLSAPR\_TRUSTED\_ENUM\_BUFFER;
255. typedef struct \_LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO {
256. RPC\_UNICODE\_STRING DomainName;
257. PRPC\_SID DomainSid;
258. } LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO, \*PLSAPR\_POLICY\_ACCOUNT\_DOM\_INFO;
259. typedef struct \_LSAPR\_POLICY\_PRIMARY\_DOM\_INFO {
260. RPC\_UNICODE\_STRING Name;
261. PRPC\_SID Sid;
262. } LSAPR\_POLICY\_PRIMARY\_DOM\_INFO, \*PLSAPR\_POLICY\_PRIMARY\_DOM\_INFO;
263. typedef struct \_LSAPR\_POLICY\_DNS\_DOMAIN\_INFO {
264. RPC\_UNICODE\_STRING Name;
265. RPC\_UNICODE\_STRING DnsDomainName;
266. RPC\_UNICODE\_STRING DnsForestName;
267. GUID DomainGuid;
268. PRPC\_SID Sid;
269. } LSAPR\_POLICY\_DNS\_DOMAIN\_INFO, \*PLSAPR\_POLICY\_DNS\_DOMAIN\_INFO;
270. typedef struct \_LSAPR\_POLICY\_PD\_ACCOUNT\_INFO {
271. RPC\_UNICODE\_STRING Name;
272. } LSAPR\_POLICY\_PD\_ACCOUNT\_INFO, \*PLSAPR\_POLICY\_PD\_ACCOUNT\_INFO;
273. typedef struct \_LSAPR\_POLICY\_REPLICA\_SRCE\_INFO {
274. RPC\_UNICODE\_STRING ReplicaSource;
275. RPC\_UNICODE\_STRING ReplicaAccountName;
276. } LSAPR\_POLICY\_REPLICA\_SRCE\_INFO, \*PLSAPR\_POLICY\_REPLICA\_SRCE\_INFO;
277. typedef struct \_LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO {
278. unsigned char AuditingMode;
279. [size\_is(MaximumAuditEventCount)]
280. unsigned long \*EventAuditingOptions;
281. [range(0,1000)] unsigned long MaximumAuditEventCount;
282. } LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO, \*PLSAPR\_POLICY\_AUDIT\_EVENTS\_INFO;
283. typedef [switch\_type(POLICY\_INFORMATION\_CLASS)]
284. union \_LSAPR\_POLICY\_INFORMATION {
285. [case(PolicyAuditLogInformation)]
286. POLICY\_AUDIT\_LOG\_INFO PolicyAuditLogInfo;
287. [case(PolicyAuditEventsInformation)]
288. LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO PolicyAuditEventsInfo;
289. [case(PolicyPrimaryDomainInformation)]
290. LSAPR\_POLICY\_PRIMARY\_DOM\_INFO PolicyPrimaryDomainInfo;
291. [case(PolicyAccountDomainInformation)]
292. LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO PolicyAccountDomainInfo;
293. [case(PolicyPdAccountInformation)]
294. LSAPR\_POLICY\_PD\_ACCOUNT\_INFO PolicyPdAccountInfo;
295. [case(PolicyLsaServerRoleInformation)]
296. POLICY\_LSA\_SERVER\_ROLE\_INFO PolicyServerRoleInfo;
297. [case(PolicyReplicaSourceInformation)]
298. LSAPR\_POLICY\_REPLICA\_SRCE\_INFO PolicyReplicaSourceInfo;
299. [case(PolicyModificationInformation)]
300. POLICY\_MODIFICATION\_INFO PolicyModificationInfo;
301. [case(PolicyAuditFullSetInformation)]
302. POLICY\_AUDIT\_FULL\_SET\_INFO PolicyAuditFullSetInfo;
303. [case(PolicyAuditFullQueryInformation)]
304. POLICY\_AUDIT\_FULL\_QUERY\_INFO PolicyAuditFullQueryInfo;
305. [case(PolicyDnsDomainInformation)]
306. LSAPR\_POLICY\_DNS\_DOMAIN\_INFO PolicyDnsDomainInfo;
307. [case(PolicyDnsDomainInformationInt)]
308. LSAPR\_POLICY\_DNS\_DOMAIN\_INFO PolicyDnsDomainInfoInt;
309. [case(PolicyLocalAccountDomainInformation)]
310. LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO PolicyLocalAccountDomainInfo;
311. } LSAPR\_POLICY\_INFORMATION, \*PLSAPR\_POLICY\_INFORMATION;
312. typedef struct \_POLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO {
313. unsigned long QualityOfService;
314. } POLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO,
315. \*PPOLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO;
316. typedef struct \_LSAPR\_POLICY\_DOMAIN\_EFS\_INFO {
317. unsigned long InfoLength;
318. [size\_is(InfoLength)] unsigned char \* EfsBlob;
319. } LSAPR\_POLICY\_DOMAIN\_EFS\_INFO, \*PLSAPR\_POLICY\_DOMAIN\_EFS\_INFO;
320. typedef [switch\_type(POLICY\_DOMAIN\_INFORMATION\_CLASS)]
321. union \_LSAPR\_POLICY\_DOMAIN\_INFORMATION {
322. [case(PolicyDomainQualityOfServiceInformation)]
323. POLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO
324. PolicyDomainQualityOfServiceInfo;
325. [case(PolicyDomainEfsInformation)]
326. LSAPR\_POLICY\_DOMAIN\_EFS\_INFO PolicyDomainEfsInfo;
327. [case(PolicyDomainKerberosTicketInformation)]
328. POLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO
329. PolicyDomainKerbTicketInfo;
330. } LSAPR\_POLICY\_DOMAIN\_INFORMATION, \*PLSAPR\_POLICY\_DOMAIN\_INFORMATION;
331. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO {
332. RPC\_UNICODE\_STRING Name;
333. } LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO, \*PLSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO;
334. typedef struct \_LSAPR\_TRUSTED\_CONTROLLERS\_INFO {
335. [range(0,5)] unsigned long Entries;
336. [size\_is(Entries)] PRPC\_UNICODE\_STRING Names;
337. } LSAPR\_TRUSTED\_CONTROLLERS\_INFO, \*PLSAPR\_TRUSTED\_CONTROLLERS\_INFO;
338. typedef struct \_LSAPR\_TRUSTED\_PASSWORD\_INFO {
339. PLSAPR\_CR\_CIPHER\_VALUE Password;
340. PLSAPR\_CR\_CIPHER\_VALUE OldPassword;
341. } LSAPR\_TRUSTED\_PASSWORD\_INFO, \*PLSAPR\_TRUSTED\_PASSWORD\_INFO;
342. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX {
343. RPC\_UNICODE\_STRING Name;
344. RPC\_UNICODE\_STRING FlatName;
345. PRPC\_SID Sid;
346. unsigned long TrustDirection;
347. unsigned long TrustType;
348. unsigned long TrustAttributes;
349. } LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX,
350. \*PLSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX;
351. typedef struct \_LSAPR\_AUTH\_INFORMATION {
352. LARGE\_INTEGER LastUpdateTime;
353. unsigned long AuthType;
354. [range(0,65536)] unsigned long AuthInfoLength;
355. [size\_is(AuthInfoLength)] unsigned char \* AuthInfo;
356. } LSAPR\_AUTH\_INFORMATION, \*PLSAPR\_AUTH\_INFORMATION;
357. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION {
358. [range(0,1)] unsigned long IncomingAuthInfos;
359. PLSAPR\_AUTH\_INFORMATION IncomingAuthenticationInformation;
360. PLSAPR\_AUTH\_INFORMATION
361. IncomingPreviousAuthenticationInformation;
362. [range(0,1)] unsigned long OutgoingAuthInfos;
363. PLSAPR\_AUTH\_INFORMATION OutgoingAuthenticationInformation;
364. PLSAPR\_AUTH\_INFORMATION
365. OutgoingPreviousAuthenticationInformation;
366. } LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION,
367. \*PLSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION;
368. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION {
369. LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX Information;
370. TRUSTED\_POSIX\_OFFSET\_INFO PosixOffset;
371. LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION AuthInformation;
372. } LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION,
373. \*PLSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION;
374. typedef LSAPR\_TRUST\_INFORMATION
375. LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_BASIC;
376. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB {
377. [range(0, 65536)] unsigned long AuthSize;
378. [size\_is( AuthSize )] unsigned char \* AuthBlob;
379. } LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB, \*PLSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB;
380. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL {
381. LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB AuthBlob;
382. } LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL,
383. \*PLSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL;
384. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL {
385. LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX Information;
386. TRUSTED\_POSIX\_OFFSET\_INFO PosixOffset;
387. LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL AuthInformation;
388. } LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL,
389. \*PLSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL;
390. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 {
391. RPC\_UNICODE\_STRING Name;
392. RPC\_UNICODE\_STRING FlatName;
393. PRPC\_SID Sid;
394. unsigned long TrustDirection;
395. unsigned long TrustType;
396. unsigned long TrustAttributes;
397. unsigned long ForestTrustLength;
398. [size\_is(ForestTrustLength)] unsigned char \* ForestTrustInfo;
399. } LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2,
400. \*PLSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2;
401. typedef struct \_LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2 {
402. LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 Information;
403. TRUSTED\_POSIX\_OFFSET\_INFO PosixOffset;
404. LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION AuthInformation;
405. } LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2,
406. \*PLSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2;
407. typedef struct \_TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES {
408. unsigned long SupportedEncryptionTypes;
409. } TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES,
410. \*PTRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES;
411. typedef [switch\_type(TRUSTED\_INFORMATION\_CLASS)]
412. union \_LSAPR\_TRUSTED\_DOMAIN\_INFO {
413. [case(TrustedDomainNameInformation)]
414. LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO TrustedDomainNameInfo;
415. [case(TrustedControllersInformation)]
416. LSAPR\_TRUSTED\_CONTROLLERS\_INFO TrustedControllersInfo;
417. [case(TrustedPosixOffsetInformation)]
418. TRUSTED\_POSIX\_OFFSET\_INFO TrustedPosixOffsetInfo;
419. [case(TrustedPasswordInformation)]
420. LSAPR\_TRUSTED\_PASSWORD\_INFO TrustedPasswordInfo;
421. [case(TrustedDomainInformationBasic)]
422. LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_BASIC TrustedDomainInfoBasic;
423. [case(TrustedDomainInformationEx)]
424. LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX TrustedDomainInfoEx;
425. [case(TrustedDomainAuthInformation)]
426. LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION TrustedAuthInfo;
427. [case(TrustedDomainFullInformation)]
428. LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION TrustedFullInfo;
429. [case(TrustedDomainAuthInformationInternal)]
430. LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL
431. TrustedAuthInfoInternal;
432. [case(TrustedDomainFullInformationInternal)]
433. LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL
434. TrustedFullInfoInternal;
435. [case(TrustedDomainInformationEx2Internal)]
436. LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 TrustedDomainInfoEx2;
437. [case(TrustedDomainFullInformation2Internal)]
438. LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2 TrustedFullInfo2;
439. [case(TrustedDomainSupportedEncryptionTypes)]
440. TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES TrustedDomainSETs;
441. } LSAPR\_TRUSTED\_DOMAIN\_INFO, \*PLSAPR\_TRUSTED\_DOMAIN\_INFO;
442. typedef struct \_LSAPR\_USER\_RIGHT\_SET {
443. [range(0,256)] unsigned long Entries;
444. [size\_is(Entries)] PRPC\_UNICODE\_STRING UserRights;
445. } LSAPR\_USER\_RIGHT\_SET, \*PLSAPR\_USER\_RIGHT\_SET;
446. typedef struct \_LSAPR\_TRUSTED\_ENUM\_BUFFER\_EX {
447. unsigned long EntriesRead;
448. [size\_is(EntriesRead)]
449. PLSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX EnumerationBuffer;
450. } LSAPR\_TRUSTED\_ENUM\_BUFFER\_EX, \*PLSAPR\_TRUSTED\_ENUM\_BUFFER\_EX;
451. //
452. // Methods
453. //
454. //
455. // The following notation conventions are used for some IDL methods:
456. //
457. // void
458. // Lsar\_LSA\_TM\_XX( void );
459. //
460. // (where XX represents the opnum.)
461. //
462. // This notation indicates that the method is defined in this
463. // interface but is described in the
464. // Local Security Authority (Translation Methods) protocol
465. // specification.
466. //
467. // void OpnumXXNotUsedOnWire(void);
468. //
469. // (where XX represents the opnum.)
470. //
471. // This notation indicates that the method is defined in this
472. // interface but is not seen on the wire.
473. //
474. // Opnum 0
475. NTSTATUS
476. LsarClose(
477. [in,out] LSAPR\_HANDLE \*ObjectHandle
478. );
479. // Opnum 1
480. void Opnum1NotUsedOnWire(void);
481. // Opnum 2
482. NTSTATUS
483. LsarEnumeratePrivileges(
484. [in] LSAPR\_HANDLE PolicyHandle,
485. [in, out] unsigned long \*EnumerationContext,
486. [out] PLSAPR\_PRIVILEGE\_ENUM\_BUFFER EnumerationBuffer,
487. [in] unsigned long PreferedMaximumLength
488. );
489. // Opnum 3
490. NTSTATUS
491. LsarQuerySecurityObject(
492. [in] LSAPR\_HANDLE ObjectHandle,
493. [in] SECURITY\_INFORMATION SecurityInformation,
494. [out] PLSAPR\_SR\_SECURITY\_DESCRIPTOR \*SecurityDescriptor
495. );
496. // Opnum 4
497. NTSTATUS
498. LsarSetSecurityObject(
499. [in] LSAPR\_HANDLE ObjectHandle,
500. [in] SECURITY\_INFORMATION SecurityInformation,
501. [in] PLSAPR\_SR\_SECURITY\_DESCRIPTOR SecurityDescriptor
502. );
503. // Opnum 5
504. void Opnum5NotUsedOnWire(void);
505. // Opnum 6
506. NTSTATUS
507. LsarOpenPolicy(
508. [in,unique] wchar\_t \*SystemName,
509. [in] PLSAPR\_OBJECT\_ATTRIBUTES ObjectAttributes,
510. [in] ACCESS\_MASK DesiredAccess,
511. [out] LSAPR\_HANDLE \*PolicyHandle
512. );
513. // Opnum 7
514. NTSTATUS
515. LsarQueryInformationPolicy(
516. [in] LSAPR\_HANDLE PolicyHandle,
517. [in] POLICY\_INFORMATION\_CLASS InformationClass,
518. [out, switch\_is(InformationClass)]
519. PLSAPR\_POLICY\_INFORMATION \*PolicyInformation
520. );
521. // Opnum 8
522. NTSTATUS
523. LsarSetInformationPolicy(
524. [in] LSAPR\_HANDLE PolicyHandle,
525. [in] POLICY\_INFORMATION\_CLASS InformationClass,
526. [in, switch\_is(InformationClass)]
527. PLSAPR\_POLICY\_INFORMATION PolicyInformation
528. );
529. // Opnum 9
530. void Opnum9NotUsedOnWire(void);
531. // Opnum 10
532. NTSTATUS
533. LsarCreateAccount(
534. [in] LSAPR\_HANDLE PolicyHandle,
535. [in] PRPC\_SID AccountSid,
536. [in] ACCESS\_MASK DesiredAccess,
537. [out] LSAPR\_HANDLE \*AccountHandle
538. );
539. // Opnum 11
540. NTSTATUS
541. LsarEnumerateAccounts(
542. [in] LSAPR\_HANDLE PolicyHandle,
543. [in] [out] unsigned long \*EnumerationContext,
544. [out] PLSAPR\_ACCOUNT\_ENUM\_BUFFER EnumerationBuffer,
545. [in] unsigned long PreferedMaximumLength
546. );
547. // Opnum 12
548. NTSTATUS
549. LsarCreateTrustedDomain(
550. [in] LSAPR\_HANDLE PolicyHandle,
551. [in] PLSAPR\_TRUST\_INFORMATION TrustedDomainInformation,
552. [in] ACCESS\_MASK DesiredAccess,
553. [out] LSAPR\_HANDLE \*TrustedDomainHandle
554. );
555. // Opnum 13
556. NTSTATUS
557. LsarEnumerateTrustedDomains(
558. [in] LSAPR\_HANDLE PolicyHandle,
559. [in] [out] unsigned long \*EnumerationContext,
560. [out] PLSAPR\_TRUSTED\_ENUM\_BUFFER EnumerationBuffer,
561. [in] unsigned long PreferedMaximumLength
562. );
563. // Opnum 14
564. void
565. Lsar\_LSA\_TM\_14( void );
566. // Opnum 15
567. void
568. Lsar\_LSA\_TM\_15( void );
569. // Opnum 16
570. NTSTATUS
571. LsarCreateSecret(
572. [in] LSAPR\_HANDLE PolicyHandle,
573. [in] PRPC\_UNICODE\_STRING SecretName,
574. [in] ACCESS\_MASK DesiredAccess,
575. [out] LSAPR\_HANDLE \*SecretHandle
576. );
577. // Opnum 17
578. NTSTATUS
579. LsarOpenAccount(
580. [in] LSAPR\_HANDLE PolicyHandle,
581. [in] PRPC\_SID AccountSid,
582. [in] ACCESS\_MASK DesiredAccess,
583. [out] LSAPR\_HANDLE \*AccountHandle
584. );
585. // Opnum 18
586. NTSTATUS
587. LsarEnumeratePrivilegesAccount(
588. [in] LSAPR\_HANDLE AccountHandle,
589. [out] PLSAPR\_PRIVILEGE\_SET \*Privileges
590. );
591. // Opnum 19
592. NTSTATUS
593. LsarAddPrivilegesToAccount(
594. [in] LSAPR\_HANDLE AccountHandle,
595. [in] PLSAPR\_PRIVILEGE\_SET Privileges
596. );
597. // Opnum 20
598. NTSTATUS
599. LsarRemovePrivilegesFromAccount(
600. [in] LSAPR\_HANDLE AccountHandle,
601. [in] unsigned char AllPrivileges,
602. [in, unique] PLSAPR\_PRIVILEGE\_SET Privileges
603. );
604. // Opnum 21
605. void Opnum21NotUsedOnWire(void);
606. // Opnum 22
607. void Opnum22NotUsedOnWire(void);
608. // Opnum 23
609. NTSTATUS
610. LsarGetSystemAccessAccount(
611. [in] LSAPR\_HANDLE AccountHandle,
612. [out] unsigned long \*SystemAccess
613. );
614. // Opnum 24
615. NTSTATUS
616. LsarSetSystemAccessAccount(
617. [in] LSAPR\_HANDLE AccountHandle,
618. [in] unsigned long SystemAccess
619. );
620. // Opnum 25
621. NTSTATUS
622. LsarOpenTrustedDomain(
623. [in] LSAPR\_HANDLE PolicyHandle,
624. [in] PRPC\_SID TrustedDomainSid,
625. [in] ACCESS\_MASK DesiredAccess,
626. [out] LSAPR\_HANDLE \*TrustedDomainHandle
627. );
628. // Opnum 26
629. NTSTATUS
630. LsarQueryInfoTrustedDomain(
631. [in] LSAPR\_HANDLE TrustedDomainHandle,
632. [in] TRUSTED\_INFORMATION\_CLASS InformationClass,
633. [out, switch\_is(InformationClass)]
634. PLSAPR\_TRUSTED\_DOMAIN\_INFO \*TrustedDomainInformation
635. );
636. // Opnum 27
637. NTSTATUS
638. LsarSetInformationTrustedDomain(
639. [in] LSAPR\_HANDLE TrustedDomainHandle,
640. [in] TRUSTED\_INFORMATION\_CLASS InformationClass,
641. [in, switch\_is(InformationClass)]
642. PLSAPR\_TRUSTED\_DOMAIN\_INFO TrustedDomainInformation
643. );
644. // Opnum 28
645. NTSTATUS
646. LsarOpenSecret(
647. [in] LSAPR\_HANDLE PolicyHandle,
648. [in] PRPC\_UNICODE\_STRING SecretName,
649. [in] ACCESS\_MASK DesiredAccess,
650. [out] LSAPR\_HANDLE \*SecretHandle
651. );
652. // Opnum 29
653. NTSTATUS
654. LsarSetSecret(
655. [in] LSAPR\_HANDLE SecretHandle,
656. [in, unique] PLSAPR\_CR\_CIPHER\_VALUE EncryptedCurrentValue,
657. [in, unique] PLSAPR\_CR\_CIPHER\_VALUE EncryptedOldValue
658. );
659. // Opnum 30
660. NTSTATUS
661. LsarQuerySecret(
662. [in] LSAPR\_HANDLE SecretHandle,
663. [in, out, unique] PLSAPR\_CR\_CIPHER\_VALUE \*EncryptedCurrentValue,
664. [in, out, unique] PLARGE\_INTEGER CurrentValueSetTime,
665. [in, out, unique] PLSAPR\_CR\_CIPHER\_VALUE \*EncryptedOldValue,
666. [in, out, unique] PLARGE\_INTEGER OldValueSetTime
667. );
668. // Opnum 31
669. NTSTATUS
670. LsarLookupPrivilegeValue(
671. [in] LSAPR\_HANDLE PolicyHandle,
672. [in] PRPC\_UNICODE\_STRING Name,
673. [out] PLUID Value
674. );
675. // Opnum 32
676. NTSTATUS
677. LsarLookupPrivilegeName(
678. [in] LSAPR\_HANDLE PolicyHandle,
679. [in] PLUID Value,
680. [out] PRPC\_UNICODE\_STRING \*Name
681. );
682. // Opnum 33
683. NTSTATUS
684. LsarLookupPrivilegeDisplayName(
685. [in] LSAPR\_HANDLE PolicyHandle,
686. [in] PRPC\_UNICODE\_STRING Name,
687. [in] short ClientLanguage,
688. [in] short ClientSystemDefaultLanguage,
689. [out] PRPC\_UNICODE\_STRING \*DisplayName,
690. [out] unsigned short \*LanguageReturned
691. );
692. // Opnum 34
693. NTSTATUS
694. LsarDeleteObject(
695. [in,out] LSAPR\_HANDLE \*ObjectHandle
696. );
697. // Opnum 35
698. NTSTATUS
699. LsarEnumerateAccountsWithUserRight(
700. [in] LSAPR\_HANDLE PolicyHandle,
701. [in,unique] PRPC\_UNICODE\_STRING UserRight,
702. [out] PLSAPR\_ACCOUNT\_ENUM\_BUFFER EnumerationBuffer
703. );
704. // Opnum 36
705. NTSTATUS
706. LsarEnumerateAccountRights(
707. [in] LSAPR\_HANDLE PolicyHandle,
708. [in] PRPC\_SID AccountSid,
709. [out] PLSAPR\_USER\_RIGHT\_SET UserRights
710. );
711. // Opnum 37
712. NTSTATUS
713. LsarAddAccountRights(
714. [in] LSAPR\_HANDLE PolicyHandle,
715. [in] PRPC\_SID AccountSid,
716. [in] PLSAPR\_USER\_RIGHT\_SET UserRights
717. );
718. // Opnum 38
719. NTSTATUS
720. LsarRemoveAccountRights(
721. [in] LSAPR\_HANDLE PolicyHandle,
722. [in] PRPC\_SID AccountSid,
723. [in] unsigned char AllRights,
724. [in] PLSAPR\_USER\_RIGHT\_SET UserRights
725. );
726. // Opnum 39
727. NTSTATUS
728. LsarQueryTrustedDomainInfo(
729. [in] LSAPR\_HANDLE PolicyHandle,
730. [in] PRPC\_SID TrustedDomainSid,
731. [in] TRUSTED\_INFORMATION\_CLASS InformationClass,
732. [out, switch\_is(InformationClass)]
733. PLSAPR\_TRUSTED\_DOMAIN\_INFO \* TrustedDomainInformation
734. );
735. // Opnum 40
736. NTSTATUS
737. LsarSetTrustedDomainInfo(
738. [in] LSAPR\_HANDLE PolicyHandle,
739. [in] PRPC\_SID TrustedDomainSid,
740. [in] TRUSTED\_INFORMATION\_CLASS InformationClass,
741. [in, switch\_is(InformationClass)]
742. PLSAPR\_TRUSTED\_DOMAIN\_INFO TrustedDomainInformation
743. );
744. // Opnum 41
745. NTSTATUS
746. LsarDeleteTrustedDomain(
747. [in] LSAPR\_HANDLE PolicyHandle,
748. [in] PRPC\_SID TrustedDomainSid
749. );
750. // Opnum 42
751. NTSTATUS
752. LsarStorePrivateData(
753. [in] LSAPR\_HANDLE PolicyHandle,
754. [in] PRPC\_UNICODE\_STRING KeyName,
755. [in,unique] PLSAPR\_CR\_CIPHER\_VALUE EncryptedData
756. );
757. // Opnum 43
758. NTSTATUS
759. LsarRetrievePrivateData(
760. [in] LSAPR\_HANDLE PolicyHandle,
761. [in] PRPC\_UNICODE\_STRING KeyName,
762. [in, out] PLSAPR\_CR\_CIPHER\_VALUE \*EncryptedData
763. );
764. // Opnum 44
765. NTSTATUS
766. LsarOpenPolicy2(
767. [in,unique,string] wchar\_t \*SystemName,
768. [in] PLSAPR\_OBJECT\_ATTRIBUTES ObjectAttributes,
769. [in] ACCESS\_MASK DesiredAccess,
770. [out] LSAPR\_HANDLE \*PolicyHandle
771. );
772. // Opnum 45
773. void
774. Lsar\_LSA\_TM\_45( void );
775. // Opnum 46
776. NTSTATUS
777. LsarQueryInformationPolicy2(
778. [in] LSAPR\_HANDLE PolicyHandle,
779. [in] POLICY\_INFORMATION\_CLASS InformationClass,
780. [out, switch\_is(InformationClass)]
781. PLSAPR\_POLICY\_INFORMATION \*PolicyInformation
782. );
783. // Opnum 47
784. NTSTATUS
785. LsarSetInformationPolicy2(
786. [in] LSAPR\_HANDLE PolicyHandle,
787. [in] POLICY\_INFORMATION\_CLASS InformationClass,
788. [in, switch\_is(InformationClass)]
789. PLSAPR\_POLICY\_INFORMATION PolicyInformation
790. );
791. // Opnum 48
792. NTSTATUS
793. LsarQueryTrustedDomainInfoByName(
794. [in] LSAPR\_HANDLE PolicyHandle,
795. [in] PRPC\_UNICODE\_STRING TrustedDomainName,
796. [in] TRUSTED\_INFORMATION\_CLASS InformationClass,
797. [out, switch\_is(InformationClass)]
798. PLSAPR\_TRUSTED\_DOMAIN\_INFO \*TrustedDomainInformation
799. );
800. // Opnum 49
801. NTSTATUS
802. LsarSetTrustedDomainInfoByName(
803. [in] LSAPR\_HANDLE PolicyHandle,
804. [in] PRPC\_UNICODE\_STRING TrustedDomainName,
805. [in] TRUSTED\_INFORMATION\_CLASS InformationClass,
806. [in, switch\_is(InformationClass)]
807. PLSAPR\_TRUSTED\_DOMAIN\_INFO TrustedDomainInformation
808. );
809. // Opnum 50
810. NTSTATUS
811. LsarEnumerateTrustedDomainsEx(
812. [in] LSAPR\_HANDLE PolicyHandle,
813. [in, out] unsigned long \*EnumerationContext,
814. [out] PLSAPR\_TRUSTED\_ENUM\_BUFFER\_EX EnumerationBuffer,
815. [in] unsigned long PreferedMaximumLength
816. );
817. // Opnum 51
818. NTSTATUS
819. LsarCreateTrustedDomainEx(
820. [in] LSAPR\_HANDLE PolicyHandle,
821. [in] PLSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX
822. TrustedDomainInformation,
823. [in] PLSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION
824. AuthenticationInformation,
825. [in] ACCESS\_MASK DesiredAccess,
826. [out] LSAPR\_HANDLE \*TrustedDomainHandle
827. );
828. // Opnum 52
829. void Opnum52NotUsedOnWire(void);
830. // Opnum 53
831. NTSTATUS
832. LsarQueryDomainInformationPolicy(
833. [in] LSAPR\_HANDLE PolicyHandle,
834. [in] POLICY\_DOMAIN\_INFORMATION\_CLASS InformationClass,
835. [out, switch\_is(InformationClass)]
836. PLSAPR\_POLICY\_DOMAIN\_INFORMATION \*PolicyDomainInformation
837. );
838. // Opnum 54
839. NTSTATUS
840. LsarSetDomainInformationPolicy(
841. [in] LSAPR\_HANDLE PolicyHandle,
842. [in] POLICY\_DOMAIN\_INFORMATION\_CLASS InformationClass,
843. [in, unique, switch\_is(InformationClass)]
844. PLSAPR\_POLICY\_DOMAIN\_INFORMATION PolicyDomainInformation
845. );
846. // Opnum 55
847. NTSTATUS
848. LsarOpenTrustedDomainByName(
849. [in] LSAPR\_HANDLE PolicyHandle,
850. [in] PRPC\_UNICODE\_STRING TrustedDomainName,
851. [in] ACCESS\_MASK DesiredAccess,
852. [out] LSAPR\_HANDLE \*TrustedDomainHandle
853. );
854. // Opnum 56
855. void Opnum56NotUsedOnWire(void);
856. // Opnum 57
857. void
858. Lsar\_LSA\_TM\_57( void );
859. // Opnum 58
860. void
861. Lsar\_LSA\_TM\_58( void );
862. // Opnum 59
863. NTSTATUS
864. LsarCreateTrustedDomainEx2(
865. [in] LSAPR\_HANDLE PolicyHandle,
866. [in] PLSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX
867. TrustedDomainInformation,
868. [in] PLSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL
869. AuthenticationInformation,
870. [in] ACCESS\_MASK DesiredAccess,
871. [out] LSAPR\_HANDLE \*TrustedDomainHandle
872. );
873. // Opnum 60
874. void Opnum60NotUsedOnWire(void);
875. // Opnum 61
876. void Opnum61NotUsedOnWire(void);
877. // Opnum 62
878. void Opnum62NotUsedOnWire(void);
879. // Opnum 63
880. void Opnum63NotUsedOnWire(void);
881. // Opnum 64
882. void Opnum64NotUsedOnWire(void);
883. // Opnum 65
884. void Opnum65NotUsedOnWire(void);
885. // Opnum 66
886. void Opnum66NotUsedOnWire(void);
887. // Opnum 67
888. void Opnum67NotUsedOnWire(void);
889. // Opnum 68
890. void
891. Lsar\_LSA\_TM\_68( void );
892. // Opnum 69
893. void Opnum69NotUsedOnWire(void);
894. // Opnum 70
895. void Opnum70NotUsedOnWire(void);
896. // Opnum 71
897. void Opnum71NotUsedOnWire(void);
898. // Opnum 72
899. void Opnum72NotUsedOnWire(void);
900. // Opnum 73
901. NTSTATUS
902. LsarQueryForestTrustInformation(
903. [in] LSAPR\_HANDLE PolicyHandle,
904. [in] PLSA\_UNICODE\_STRING TrustedDomainName,
905. [in] LSA\_FOREST\_TRUST\_RECORD\_TYPE HighestRecordType,
906. [out] PLSA\_FOREST\_TRUST\_INFORMATION \* ForestTrustInfo
907. );
908. // Opnum 74
909. NTSTATUS
910. LsarSetForestTrustInformation(
911. [in] LSAPR\_HANDLE PolicyHandle,
912. [in] PLSA\_UNICODE\_STRING TrustedDomainName,
913. [in] LSA\_FOREST\_TRUST\_RECORD\_TYPE HighestRecordType,
914. [in] PLSA\_FOREST\_TRUST\_INFORMATION ForestTrustInfo,
915. [in] unsigned char CheckOnly,
916. [out] PLSA\_FOREST\_TRUST\_COLLISION\_INFORMATION \* CollisionInfo
917. );
918. }

# Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

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

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 NT operating system | Yes | Yes |
| 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 NT | Yes | Yes |
| Windows 2000 Server operating system | Yes | Yes |
| Windows Server 2003 operating system | Yes | Yes |
| Windows Server 2003 for Small Business Server 2003 | 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): By default, the "\PIPE\lsarpc" [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) allows anonymous access on Windows NT 3.1 operating system, Windows NT 3.5 operating system, Windows NT 3.51 operating system, Windows NT 4.0 operating system, Windows 2000 operating system, Windows XP, Windows Server 2003, Windows Server 2003 R2, and Windows Vista RTM. Anonymous access to this pipe is removed by default on Windows Vista operating system with Service Pack 1 (SP1) and later and Windows Server 2008 and later in both the non-[**domain controller**](#gt_76a05049-3531-4abd-aec8-30e19954b4bd) configuration and the [**read-only domain controller**](#gt_8b0a073b-3099-4efe-8b81-c2886b66a870) configuration. The pipe access check happens before any other access check; therefore, it overrides any other access.

[<2> Section 2.1](#Appendix_A_Target_2): Windows implementations of the client and [**server role**](#gt_585b8697-30d5-4431-be5b-5a78e4a63c20) for this protocol use the tamper-resistance functionality provided by [**SMB**](#gt_09dbec39-5e75-4d9a-babf-1c9f1d499625) transport on the products that are available, and are enabled as specified in [[MS-SMB]](%5bMS-SMB%5d.pdf#Section_f210069c70864dc2885e861d837df688) section 3.1.1.1 (the *MessageSigningPolicy* parameter), and [[MS-SMB2]](%5bMS-SMB2%5d.pdf#Section_5606ad475ee0437a817e70c366052962) section 3.1.1.1 (the *RequireMessageSigning* parameter).

[<3> Section 2.1](#Appendix_A_Target_3): If an implementation of the client role violates this specification and uses the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331)-provided security-support-provider mechanism for the RPC connection to a Windows implementation, Windows processes all messages as specified in section [3.1](#Section_63acb76d473a407dbc0c548118e53055) (that is, there is no change in message processing behavior), except for the messages that use encryption specified in section [5.1](#Section_ed292c30f14a4568b5992e69358144d8). During encryption and decryption, Windows implementations for the server role use a hard-coded key instead of the SMB transport–provided session key. The hard-coded key is represented below as bytes in hexadecimal form.

"53 79 73 74 65 6d 4c 69-62 72 61 72 79 44 54 43"

[<4> Section 2.1](#Appendix_A_Target_4): The Windows implementation of the server role for this protocol supports the RPC-provided security-support-provider mechanisms, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.2.1.4.1. The following security-support providers are registered by the responder.

| Windows version | Security support provider registered |
| --- | --- |
| Windows NT and Windows 2000 Professional and later | RPC\_C\_AUTHN\_WINNT |
| Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later | RPC\_C\_AUTHN\_WINNT  On the domain controllers the following are also supported:  RPC\_C\_AUTHN\_GSS\_KERBEROS  RPC\_C\_AUTHN\_GSS\_NEGOTIATE |

[<5> Section 2.1](#Appendix_A_Target_5): Servers running Windows 2000, Windows XP, and Windows Server 2003 accept calls at any authentication level. Without [[MSKB-3149090]](https://go.microsoft.com/fwlink/?LinkId=786963) installed, servers running Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, Windows Server 2012 R2, Windows 10 v1507 operating system, or Windows 10 v1511 operating system also accept calls at any authentication level.

[<6> Section 2.1](#Appendix_A_Target_6): The server implementation of this protocol in Windows 2000 and earlier does not enforce a limit. The limit in Windows XP and Windows Server 2003 is 4 MB.

[<7> Section 2.2](#Appendix_A_Target_7): Data type fields that are described as "Reserved" or "MUST be ignored" are sent as 0 (or NULL in the case of pointers) by the Windows implementation of the protocol client, and are ignored upon receipt by the Windows implementation of the protocol server.

[<8> Section 2.2](#Appendix_A_Target_8): The following table is a timeline of when each structure, data type, or enumeration was introduced. All structures, data types, and enumerations listed in the table continue to be available in subsequent versions of Windows according to the applicability lists at the beginning of this section.

| Data type | Product |
| --- | --- |
| [LSAPR\_HANDLE (section 2.2.2.1)](#Section_0d093105e8c845f7a79d182aafd60c6e) | Windows NT 3.1 |
| [STRING (section 2.2.3.1)](#Section_94a41a4fbd5d4c3eafd4cc17e83a6e01) | Windows NT 3.1 |
| [LSAPR\_ACL (section 2.2.3.2)](#Section_a9a03a855b084bb581c92c68751693ac) | Windows NT 3.1 |
| [SECURITY\_DESCRIPTOR\_CONTROL (section 2.2.3.3)](#Section_c704a67c983641d99b18acd596cc884e) | Windows NT 3.1 |
| [LSAPR\_SECURITY\_DESCRIPTOR (section 2.2.3.4)](#Section_d5cf869d674449cca67730ccb9217def) | Windows NT 3.1 |
| [SECURITY\_IMPERSONATION\_LEVEL (section 2.2.3.5)](#Section_720cea10cee24c459084c6fa7d67d18d) | Windows NT 3.1 |
| [SECURITY\_CONTEXT\_TRACKING\_MODE (section 2.2.3.6)](#Section_6bb42770b92441ff8a5783e37b8b7797) | Windows NT 3.1 |
| [SECURITY\_QUALITY\_OF\_SERVICE (section 2.2.3.7)](#Section_0ddf315053b542a5b0ec518bce67738c) | Windows NT 3.1 |
| [LSAPR\_OBJECT\_ATTRIBUTES (section 2.2.2.4)](#Section_ad9e183d64744641a6d9d3796d2d604b) | Windows NT 3.1 |
| [ACCESS\_MASK (section 2.2.1.1)](#Section_7aeb7f170a6e4f04ac7e7b1363cf9ecf) | Windows NT 3.1 |
| [SECURITY\_INFORMATION (section 2.2.1.3)](#Section_62175da4e30f4c12b1c4dae0434e38af) | Windows NT 3.1 |
| [LSAPR\_POLICY\_PRIVILEGE\_DEF (section 2.2.8.1)](#Section_f36d47375b2f4bc08f29e7b4c71b7401) | Windows NT 3.1 |
| [LSAPR\_PRIVILEGE\_ENUM\_BUFFER (section 2.2.8.2)](#Section_c0278280b4b64538b3aaeb40f64f42fb) | Windows NT 3.1 |
| [LSAPR\_ACCOUNT\_INFORMATION (section 2.2.5.1)](#Section_98540c1c09cc4ee2934acdde3de0c77f) | Windows NT 3.1 |
| [LSAPR\_ACCOUNT\_ENUM\_BUFFER (section 2.2.5.2)](#Section_727c2d44879448969fba5e1725bc288e) | Windows NT 3.1 |
| [POLICY\_SYSTEM\_ACCESS\_CODE (section 2.2.1.2)](#Section_ba5a83c10ffe4a819e915739274c03db) | Windows NT 3.1 |
| [LSA\_UNICODE\_STRING (section 2.2.2.3)](#Section_4b35e17e405c4e998ebe8b28f047156f) | Windows NT 3.1 |
| [LSAPR\_TRUST\_INFORMATION (section 2.2.7.1)](#Section_71e86cddae194a029179a2a103b383a0) | Windows NT 3.1 |
| [LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_BASIC (section 2.2.7.8)](#Section_c101591de1b042bd8cc5f8866c3b5757) | Windows NT 3.1 |
| [LSAPR\_SR\_SECURITY\_DESCRIPTOR (section 2.2.2.5)](#Section_5564065e3f3d4481a385367cc9b042c4) | Windows NT 3.1 |
| [POLICY\_INFORMATION\_CLASS (section 2.2.4.1)](#Section_9ce0bb37fc6c4230b1097e1881660b83) | Windows NT 3.1 |
| [POLICY\_AUDIT\_LOG\_INFO (section 2.2.4.3)](#Section_3fff1c62e8b14bc8b18a3ba6458ec622) | Windows NT 3.1 |
| [LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO (section 2.2.4.4)](#Section_d00fc364577d4ed0b3a5952d78b67695) | Windows NT 3.1 |
| [LSAPR\_POLICY\_PRIMARY\_DOM\_INFO (section 2.2.4.5)](#Section_0f3f5d3f66d245a08c28ede86f4cd4a8) | Windows NT 3.1 |
| [LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO (section 2.2.4.6)](#Section_bfad54243e2043bd87f6d35b4253792e) | Windows NT 3.1 |
| [LSAPR\_POLICY\_PD\_ACCOUNT\_INFO (section 2.2.4.7)](#Section_b04175b3fedf4dda9034f754a10fe64e) | Windows NT 3.1 |
| [POLICY\_LSA\_SERVER\_ROLE (section 2.2.4.8)](#Section_620010b4b4394d46893acb67246de5fc) | Windows NT 3.1 |
| [POLICY\_LSA\_SERVER\_ROLE\_INFO (section 2.2.4.9)](#Section_d37dbc6504f34db8b40a4e9dd6c12520) | Windows NT 3.1 |
| [LSAPR\_POLICY\_REPLICA\_SRCE\_INFO (section 2.2.4.10)](#Section_fb7df2bb99e7402f833424d47e23ec00) | Windows NT 3.1 |
| [POLICY\_MODIFICATION\_INFO (section 2.2.4.11)](#Section_c80ae9d5d0c14d5ca0ae77eae7bfac25) | Windows NT 3.1 |
| [POLICY\_AUDIT\_FULL\_SET\_INFO (section 2.2.4.12)](#Section_3224400e3c404e64810a8b11341ba4c6) | Windows NT 3.1 |
| [POLICY\_AUDIT\_FULL\_QUERY\_INFO (section 2.2.4.13)](#Section_0ef0845ff20e4897ad2988c0c07be0f4) | Windows NT 3.1 |
| [LSAPR\_POLICY\_DNS\_DOMAIN\_INFO (section 2.2.4.14)](#Section_3e15a02e25d346aa9c608def03c824d2) | Windows NT 3.1 |
| [LSAPR\_POLICY\_INFORMATION (section 2.2.4.2)](#Section_6e63a2c85ddb411aa2539c55afc49834) | Windows 2000 |
| [LSAPR\_TRUSTED\_ENUM\_BUFFER (section 2.2.7.19)](#Section_78f8a2e44f3d40f5bdd19dacdf1c832c) | Windows NT 3.1 |
| [LSAPR\_PRIVILEGE\_SET (section 2.2.5.5)](#Section_a30a5720778442f4b03ab14f4e486bae) | Windows NT 3.1 |
| [TRUSTED\_INFORMATION\_CLASS (section 2.2.7.2)](#Section_360691136c3845e8920e17f8ef36f578) | Windows NT 3.1 |
| [LSAPR\_TRUSTED\_DOMAIN\_INFO (section 2.2.7.3)](#Section_65564571dd0d49a98a2a6dba8ab57091) | Windows NT 3.1 |
| [LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO (section 2.2.7.4)](#Section_71c5724f447e452c9cb9a0fd90d88594) | Windows NT 3.1 |
| [LSAPR\_TRUSTED\_CONTROLLERS\_INFO (section 2.2.7.5)](#Section_5382bd8969c646f2beb17b70e5befbc5) | Windows NT 3.1 |
| [TRUSTED\_POSIX\_OFFSET\_INFO (section 2.2.7.6)](#Section_b091ee7ef5c34b4885671b08ea002221) | Windows NT 3.1 |
| [LSAPR\_TRUSTED\_PASSWORD\_INFO (section 2.2.7.7)](#Section_33d7a9e4c9ca40219627337d89e656a3) | Windows NT 3.1 |
| [LSAPR\_CR\_CIPHER\_VALUE (section 2.2.6.1)](#Section_782eda77b82e413487c9eb5e67f18f06) | Windows NT 3.51 |
| [LSAPR\_USER\_RIGHT\_SET (section 2.2.5.3)](#Section_dcaca8ef34a342dd85b698363eb108ff) | Windows NT 3.1 |
| [POLICY\_DOMAIN\_INFORMATION\_CLASS (section 2.2.4.15)](#Section_566a61fc2e9947c899ca62f7e22cb15d) | Windows NT 3.51 |
| [LSAPR\_POLICY\_DOMAIN\_INFORMATION (section 2.2.4.16)](#Section_1a9c523ba67a485f8f8b8fca05ca9334) | Windows 2000 |
| [LSAPR\_POLICY\_DOMAIN\_EFS\_INFO (section 2.2.4.18)](#Section_3ba6e751cf914d87a74c488bb927a54c) | Windows 2000 |
| [LSAPR\_DOMAIN\_KERBEROS\_TICKET\_INFO (section 2.2.4.19)](#Section_afcc492012d348e0ab95a8989ebbd41d) | Windows 2000 |
| [LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX (section 2.2.7.9)](#Section_f28f42b7173c4cda98093fe4a5213ab3) | Windows 2000 |
| [LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 (section 2.2.7.10)](#Section_dd92d4d9227f4ef1b42bef3f056f8aaa) | Windows 2000 |
| [LSAPR\_AUTH\_INFORMATION (section 2.2.7.17)](#Section_cedb0d1bc7c0448099fc279b06f22a0c) | Windows XP and Windows Server 2003 |
| [LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION (section 2.2.7.11)](#Section_084fdb6b5bc349129aed0257159996dd) | Windows 2000 |
| [LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB (section 2.2.7.16)](#Section_da8f32a10a164194810d06cc0698595a) | Windows 2000 |
| [LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL (section 2.2.7.12)](#Section_3b1c61fe6f074d83af543a381de5c5d1) | Windows 2000 |
| [LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION (section 2.2.7.13)](#Section_9f9feebce9e141c18c4802f83a227a14) | Windows 2000 |
| [LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL (section 2.2.7.14)](#Section_2e9e2c847b004fb18de588d4cfedd2b3) | Windows 2000 |
| [LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2 (section 2.2.7.15)](#Section_e529d0945de44738adc4efa1a7d1106f) | Windows XP and Windows Server 2003 |
| LUID ([[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.3.7) | Windows NT 3.1 |
| [TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES (section 2.2.7.18)](#Section_7c519a643dc14be6a17d76817cff6e39) | Windows Vista and Windows Server 2008 |
| [LSAPR\_LUID\_AND\_ATTRIBUTES (section 2.2.5.4)](#Section_03c834c0f3104e0c832eb6e7688364d1) | Windows NT 3.1 |
| [LSA\_FOREST\_TRUST\_RECORD\_TYPE (section 2.2.7.22)](#Section_700a91e8a1a44e1b9ad6096b3cf0bef0) | Windows XP and Windows Server 2003 |
| [LSA\_FOREST\_TRUST\_BINARY\_DATA (section 2.2.7.23)](#Section_d4859b44b7764237baa112dc28c19634) | Windows XP and Windows Server 2003 |
| [LSA\_FOREST\_TRUST\_DOMAIN\_INFO (section 2.2.7.24)](#Section_451ac72fe9ad4a4f961fd04a2a5b1515) | Windows XP and Windows Server 2003 |
| [LSA\_FOREST\_TRUST\_RECORD (section 2.2.7.21)](#Section_08cf1a65ef7c46d3aa4d558f5135df3d) | Windows XP and Windows Server 2003 |
| [LSA\_FOREST\_TRUST\_INFORMATION (section 2.2.7.25)](#Section_2993ffabc0c846439a794ff7d31922dc) | Windows XP and Windows Server 2003 |
| [LSA\_FOREST\_TRUST\_COLLISION\_RECORD\_TYPE (section 2.2.7.26)](#Section_afc7d769a31748059f4585d5393b57af) | Windows XP and Windows Server 2003 |
| [LSA\_FOREST\_TRUST\_COLLISION\_RECORD (section 2.2.7.27)](#Section_32178d2cca744f538264af1906f95011) | Windows XP and Windows Server 2003 |
| [LSA\_FOREST\_TRUST\_COLLISION\_INFORMATION (section 2.2.7.28)](#Section_db0e91319a194cb5937a0e3a5767a0b2) | Windows XP and Windows Server 2003 |

[<9> Section 2.2.1.1.2](#Appendix_A_Target_9): The following is a timeline of when each access mask was introduced. All access masks continue to be available in subsequent versions of Windows according to the applicability lists at the beginning of this section.

| Value | Product |
| --- | --- |
| 0x00000000 | Windows NT 3.1 |
| POLICY\_VIEW\_LOCAL\_INFORMATION  0x00000001 | Windows NT 3.1 |
| POLICY\_VIEW\_AUDIT\_INFORMATION  0x00000002 | Windows NT 3.1 |
| POLICY\_GET\_PRIVATE\_INFORMATION  0x00000004 | Windows NT 3.1 |
| POLICY\_TRUST\_ADMIN  0x00000008 | Windows NT 3.1 |
| POLICY\_CREATE\_ACCOUNT  0x00000010 | Windows NT 3.1 |
| POLICY\_CREATE\_SECRET  0x00000020 | Windows NT 3.1 |
| POLICY\_CREATE\_PRIVILEGE  0x00000040 | Windows NT 3.1 |
| POLICY\_SET\_DEFAULT\_QUOTA\_LIMITS  0x00000080 | Windows NT 3.1 |
| POLICY\_SET\_AUDIT\_REQUIREMENTS  0x00000100 | Windows NT 3.1 |
| POLICY\_AUDIT\_LOG\_ADMIN  0x00000200 | Windows NT 3.1 |
| POLICY\_SERVER\_ADMIN  0x00000400 | Windows NT 3.1 |
| POLICY\_LOOKUP\_NAMES  0x00000800 | Windows NT 3.1 |
| POLICY\_NOTIFICATION  0x00001000 | Windows 2000 |

[<10> Section 2.2.1.1.5](#Appendix_A_Target_10): The following is a timeline of when each access mask was introduced. All access masks continue to be available in subsequent versions of Windows according to the applicability lists at the beginning of this section.

| Value | Product |
| --- | --- |
| TRUSTED\_QUERY\_DOMAIN\_NAME  0x00000001 | Windows NT 3.1 |
| TRUSTED\_QUERY\_CONTROLLERS  0x00000002 | Windows NT 3.1 |
| TRUSTED\_SET\_CONTROLLERS  0x00000004 | Windows NT 3.1 |
| TRUSTED\_QUERY\_POSIX  0x00000008 | Windows NT 3.1 |
| TRUSTED\_SET\_POSIX  0x00000010 | Windows NT 3.1 |
| TRUSTED\_SET\_AUTH  0x00000020 | Windows 2000 |
| TRUSTED\_QUERY\_AUTH  0x00000040 | Windows 2000 |

[<11> Section 2.2.1.2](#Appendix_A_Target_11): The POLICY\_MODE\_ALL flag applies to Windows 2000 and later.

[<12> Section 2.2.1.2](#Appendix_A_Target_12): The POLICY\_MODE\_ALL\_NT4 flag applies to Windows NT 3.1 through Windows NT 4.0.

[<13> Section 2.2.1.2](#Appendix_A_Target_13): The following is a timeline of when each mode was introduced. All modes continue to be available in subsequent versions of Windows according to the applicability lists at the beginning of this section.

| Value | Product |
| --- | --- |
| 0x00000000  No access | Windows NT 3.1 |
| 0x00000001  POLICY\_MODE\_INTERACTIVE | Windows NT 3.1 |
| 0x00000002  POLICY\_MODE\_NETWORK | Windows NT 3.1 |
| 0x00000004  POLICY\_MODE\_BATCH | Windows NT 3.1 |
| 0x00000010  POLICY\_MODE\_SERVICE | Windows NT 3.1 |
| 0x00000020  POLICY\_MODE\_PROXY | Windows NT 3.1 |
| 0x00000040  POLICY\_MODE\_DENY\_INTERACTIVE | Windows 2000 |
| 0x00000080  POLICY\_MODE\_DENY\_NETWORK | Windows 2000 |
| 0x00000100  POLICY\_MODE\_DENY\_BATCH | Windows 2000 |
| 0x00000200  POLICY\_MODE\_DENY\_SERVICE | Windows 2000 |
| 0x00000400  POLICY\_MODE\_REMOTE\_INTERACTIVE | Windows XP and Windows Server 2003 |
| 0x00000800  POLICY\_MODE\_DENY\_REMOTE\_INTERACTIVE | Windows XP and Windows Server 2003 |

[<14> Section 2.2.2.4](#Appendix_A_Target_14): The Windows implementation of the [**RPC client**](#gt_e5a7be6b-98db-4e8d-8116-5893f43ab48b) for this protocol leaves this structure to be filled by a higher-layer application and does not verify the structure's contents except for **RootDirectory**, which must be NULL.

[<15> Section 2.2.2.5](#Appendix_A_Target_15): In Windows NT, Windows 2000, Windows XP, and Windows XP operating system Service Pack 1 (SP1), the Windows [**RPC server**](#gt_ae65dac0-cd24-4e83-a946-6d1097b71553) and RPC client do not enforce restrictions on the **Length** field of this structure (using the range primitive specified in [MS-RPCE]).

[<16> Section 2.2.4.1](#Appendix_A_Target_16): The following is a timeline of when each enumeration value was introduced. All enumeration values continue to be available in subsequent versions of Windows according to the applicability lists at the beginning of this section.

| Value | Product |
| --- | --- |
| PolicyAuditLogInformation | Windows NT 3.1 |
| PolicyAuditEventsInformation | Windows NT 3.1 |
| PolicyPrimaryDomainInformation | Windows NT 3.1 |
| PolicyPdAccountInformation | Windows NT 3.1 |
| PolicyAccountDomainInformation | Windows NT 3.1 |
| PolicyLsaServerRoleInformation | Windows NT 3.1 |
| PolicyReplicaSourceInformation | Windows NT 3.1 |
| PolicyInformationNotUsedOnWire | Windows NT 3.1 |
| PolicyModificationInformation | Windows NT 3.1 |
| PolicyAuditFullSetInformation | Windows NT 3.1 |
| PolicyAuditFullQueryInformation | Windows NT 3.1 |
| PolicyDnsDomainInformation | Windows 2000 |
| PolicyDnsDomainInformationInt | Windows 2000 |
| PolicyLocalAccountDomainInformation | Windows Vista and Windows Server 2008 |

[<17> Section 2.2.4.4](#Appendix_A_Target_17): In Windows NT, Windows 2000, Windows XP, and Windows XP SP1, the Windows RPC server and RPC client do not enforce restrictions on the **MaximumAuditEventCount** field of this structure (using the range primitive, as specified in [MS-RPCE]).

[<18> Section 2.2.4.14](#Appendix_A_Target_18): The following applies to Windows 2000 Professional and later and to Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later.

The Windows RPC server always throws an RPC\_S\_PROCNUM\_OUT\_OF\_RANGE exception for the message processing of LsarQueryInformationPolicy, LsarQueryInformationPolicy2, LsarSetInformationPolicy, and LsarSetInformationPolicy2, if the server is configured to emulate Windows NT 4.0 for PolicyDnsDomainInformation information level.

[<19> Section 2.2.4.16](#Appendix_A_Target_19): The PolicyDomainQualityOfServiceInformation enumeration value and corresponding [POLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO](#Section_38bd52a04514468fb342d7421a51a316) structure are parts of LSAPR\_POLICY\_DOMAIN\_INFORMATION only in the Windows 2000 Server implementation of this protocol.

[<20> Section 2.2.4.18](#Appendix_A_Target_20): Microsoft implementations of the Local Security Authority (Domain Policy) Remote Protocol do not enforce data in **EfsBlob** to conform to the layout specified in [[MS-GPEF]](%5bMS-GPEF%5d.pdf#Section_14d3fd83753741a2af398e52c19ef0e3) section 2.2.1.2.1.

[<21> Section 2.2.5.3](#Appendix_A_Target_21): In Windows NT, Windows 2000, Windows XP, and Windows XP SP1, the Windows RPC server and RPC client do not enforce restrictions on the **Entries** field of this structure (using the range primitive defined in [MS-RPCE]).

[<22> Section 2.2.5.5](#Appendix_A_Target_22): In Windows NT, Windows 2000, Windows XP, and Windows XP SP1, the Windows RPC server and RPC client do not enforce restrictions on the **PrivilegeCount** field of this structure (using the range primitive specified in [MS-RPCE]).

[<23> Section 2.2.6.1](#Appendix_A_Target_23): In Windows NT, Windows 2000, Windows XP, and Windows XP SP1, the Windows RPC server and RPC client do not enforce restrictions on the **Length** field of this structure (using the range primitive as specified in [MS-RPCE]).

[<24> Section 2.2.6.1](#Appendix_A_Target_24): In Windows NT, Windows 2000, Windows XP, and Windows XP SP1, the Windows RPC server and RPC client do not enforce restrictions on the **MaximumLength** field of this structure (using the range primitive defined in [MS-RPCE]).

[<25> Section 2.2.7.2](#Appendix_A_Target_25): The following is a timeline of when each enumeration value was introduced. All enumeration values continue to be available in subsequent versions of Windows according to the applicability lists at the beginning of this section.

| Value | Product |
| --- | --- |
| TrustedDomainNameInformation | Windows NT 3.1 |
| TrustedControllersInformation | Windows NT 3.1 |
| TrustedPosixOffsetInformation | Windows NT 3.1 |
| TrustedPasswordInformation | Windows NT 3.51 |
| TrustedDomainInformationBasic | Windows 2000 |
| TrustedDomainInformationEx | Windows 2000 |
| TrustedDomainAuthInformation | Windows 2000 |
| TrustedDomainFullInformation | Windows 2000 |
| TrustedDomainAuthInformationInternal | Windows 2000 |
| TrustedDomainFullInformationInternal | Windows 2000 |
| TrustedDomainInformationEx2Internal | Windows XP and Windows Server 2003 |
| TrustedDomainFullInformation2Internal | Windows XP and Windows Server 2003 |
| TrustedDomainSupportedEncryptionTypes | Windows Vista and Windows Server 2008 |

[<26> Section 2.2.7.5](#Appendix_A_Target_26): In Windows NT, Windows 2000, Windows XP, and Windows XP SP1, the Windows RPC server and RPC client do not enforce restrictions on the **Entries** field of this structure (using the range primitive defined in [MS-RPCE]).

[<27> Section 2.2.7.9](#Appendix_A_Target_27): The following is a timeline of when each flag value was introduced. Unless otherwise specified, all flag values continue to be available in subsequent versions of Windows according to the applicability lists at the beginning of this section.

| Possible value | Value | Product |
| --- | --- | --- |
| TANT (TRUST\_ATTRIBUTE\_NON\_TRANSITIVE) | 0x00000001 | Windows 2000 |
| TAUO (TRUST\_ATTRIBUTE\_UPLEVEL\_ONLY) | 0x00000002 | Windows 2000 |
| TAQD (TRUST\_ATTRIBUTE\_QUARANTINED\_DOMAIN) | 0x00000004 | Windows 2000 operating system Service Pack 2 (SP2) and Windows XP |
| TAFT (TRUST\_ATTRIBUTE\_FOREST\_TRANSITIVE) | 0x00000008 | Windows XP and Windows Server 2003 |
| TACO (TRUST\_ATTRIBUTE\_CROSS\_ORGANIZATION) | 0x00000010 | Windows Server 2003 and Windows Vista |
| TAWF (TRUST\_ATTRIBUTE\_WITHIN\_FOREST) | 0x00000020 | Windows Server 2003 and Windows Vista |
| TATE (TRUST\_ATTRIBUTE\_TREAT\_AS\_EXTERNAL) | 0x00000040 | Windows Server 2003 and Windows Vista |
| TANC (TRUST\_ATTRIBUTE\_CROSS\_ORGANIZATION\_NO\_TGT\_DELEGATION) | 0x00000200 | Windows 8 and Windows Server 2012 operating system |
| TAPT (TRUST\_ATTRIBUTE\_PIM\_TRUST) | 0x00000400 | Windows 10 and Windows Server 2016  (Also supported on Windows 8.1 and Windows Server 2012 R2 if [[MSKB-3155495]](https://go.microsoft.com/fwlink/?LinkId=808755) is installed.) |
| Obsolete | 0x00400000 | Introduced in Windows 2000 RTM. Became obsolete in Windows 2000 operating system Service Pack 4 (SP4). |
| Obsolete | 0x00800000 | Introduced in Windows 2000 RTM. Became obsolete in Windows 2000 SP4. |

[<28> Section 2.2.7.11](#Appendix_A_Target_28): In Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, and Windows Server 2008, the Windows RPC server and RPC client do not enforce restrictions on the **IncomingAuthInfos** field of this structure (using the range primitive defined in [MS-RPCE]).

[<29> Section 2.2.7.11](#Appendix_A_Target_29): In Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, and Windows Server 2008, the Windows RPC server and RPC client do not enforce restrictions on the **OutgoingAuthInfos** field of this structure (using the range primitive defined in [MS-RPCE]).

[<30> Section 2.2.7.16](#Appendix_A_Target_30): In Windows NT, Windows 2000, Windows XP, and Windows XP SP1, the Windows RPC server and RPC client do not enforce restrictions on the **AuthSize** field of this structure (using the range primitive defined in [MS-RPCE]).

[<31> Section 2.2.7.17](#Appendix_A_Target_31): In Windows NT, Windows 2000, Windows XP, and Windows XP SP1, the Windows RPC server and RPC client do not enforce restrictions on the **AuthInfoLength** field of this structure (using the range primitive defined in [MS-RPCE]).

[<32> Section 2.2.7.23](#Appendix_A_Target_32): In Windows NT, Windows 2000, Windows XP, and Windows XP SP1, the Windows RPC server and RPC client do not enforce restrictions on the **Length** field of this structure (using the range primitive defined in [MS-RPCE]).

[<33> Section 2.2.7.25](#Appendix_A_Target_33): In Windows NT, Windows 2000, Windows XP, and Windows XP SP1, the Windows RPC server and RPC client do not enforce restrictions on the **RecordCount** field of this structure (using the range primitive defined in [MS-RPCE]).

[<34> Section 3.1.1.1](#Appendix_A_Target_34): A Windows responder for this protocol contains the following values for the policy object after setup.

| Name | Value |
| --- | --- |
| Auditing Log Information | Windows maintains the following hard-coded information about the state of the audit log:  MaximumLogSize = 8192  AuditLogPercentFull = 0  AuditRetentionPeriod = 8533315  AuditLogFullShutdownInProgress = FALSE  TimeToShutdown = 288342  NextAuditRecordId = 0 |
| Audit Full Information | Windows XP and later, and Windows Server 2003 and Windows Server 2003 R2 and later return STATUS\_INVALID\_PARAMETER for this information class. |
| Event Auditing Options | On Windows 2000 and Windows XP:  AuditingMode = FALSE  MaximumAuditEventCount = 9  EventAuditingOptions = { 0, 0, 0, 0, 0, 0, 0, 0, 0 }  On Windows Server 2003 and Windows Server 2003 R2:  AuditingMode = TRUE  MaximumAuditEventCount = 9  EventAuditingOptions = { 0, 1, 0, 0, 0, 0, 0, 0, 1 }  On Windows Vista and later and Windows Server 2008 and later:  AuditingMode = TRUE  MaximumAuditEventCount = 9  EventAuditingOptions = { 0, 0, 0, 0, 0, 0, 0, 0, 0 } |
| Primary Domain Information | Name = <Workgroup Name>  Sid = NULL |
| DNS Domain Information | Name = <Workgroup Name>  DnsDomainName = <Empty String>  DnsForestName = <Empty String>  DomainGuid = { 0 }  Sid = NULL |
| Account Domain Information | DomainName = <Machine Netbios name> DomainSid = < S-1-5-21-X-Y-Z> where X, Y, Z are random numbers |
| Server Role Information | LsaServerRole = PolicyServerRolePrimary |
| Replica Source Information | ReplicaSource=<Empty String>  ReplicaAccountName=<Empty String> |
| Kerberos Policy Information | <No value> |
| Encrypting File System (EFS) Policy Information | <No value> |
| Security Descriptor | The [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) in Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows NT 4.0, and Windows 2000 can be expressed in Security Description Definition Language (SDDL), as specified in [MS-DTYP] section 2.5.1, as follows:  O:BAG:SYD:(A;;GA;;;BA)(A;;GX;;;WD)  In Windows XP and in Windows Server 2003 and Windows Server 2003 R2 and later, the security descriptor can be expressed in SDDL as follows:  O:BAG:SYD:(A;;GA;;;BA)(A;;GX;;;WD)(A;;0x0000801;;;AN)(A;;0x00001000;;;LS)(A;;0x00001000;;;NS)  In Windows Vista and later, the security descriptor can be expressed in SDDL as follows:  O:BAG:SYD:(A;;GA;;;BA)(A;;GX;;;WD)(A;;0x0000801;;;AN)(A;;0x00001000;;;LS)(A;;0x00001000;;;NS)(A;;0x00001000;;;S-1-5-17)  See sections [2.2.1.1.1](#Section_5EE8DB785F0E47B2ABA78447FF454E3B) and [2.2.1.1.2](#Section_B61B7268987A420B84F96C75F8DC8558) for the definitions of the generic and object-specific access rights, respectively, that are included in these security descriptors. |

[<35> Section 3.1.1.1](#Appendix_A_Target_35): Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not store this information.

[<36> Section 3.1.1.1](#Appendix_A_Target_36): Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not store this information.

[<37> Section 3.1.1.1](#Appendix_A_Target_37): Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 do not store this information.

[<38> Section 3.1.1.1](#Appendix_A_Target_38): Only the Windows 2000 implementation of this protocol stores quality of [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) information.

[<39> Section 3.1.1.1](#Appendix_A_Target_39): The security descriptor in Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, Windows NT 4.0, and Windows 2000 can be expressed in Security Description Definition Language (SDDL), as specified in [MS-DTYP] section 2.5.1, as follows:

O:BAG:SYD:(A;;GA;;;BA)(A;;GX;;;WD)

In Windows XP, Windows Server 2003, and Windows Server 2003 R2, the security descriptor can be expressed in SDDL as follows:

O:BAG:SYD:(A;;GA;;;BA)(A;;GX;;;WD)(A;;0x0000801;;;AN)(A;;0x00001000;;;LS) (A;;0x00001000;;;NS)

In Windows Vista and later and in Windows Server 2008 and later, the security descriptor can be expressed in SDDL as follows:

O:BAG:SYD:(A;;GA;;;BA)(A;;GX;;;WD)(A;;0x0000801;;;AN)(A;;0x00001000;;;LS) (A;;0x00001000;;;NS) (A;;0x00001000;;;S-1-5-17)

See sections 2.2.1.1.1 and 2.2.1.1.2 for the definitions of the generic and object-specific access rights, respectively, that are included in these security descriptors.

[<40> Section 3.1.1.1](#Appendix_A_Target_40): Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 domain controllers use the Netlogon Remote Protocol, as specified in [[MS-NRPC]](%5bMS-NRPC%5d.pdf#Section_ff8f970f3e3740f7bd4baf7336e4792f) section 1.3.3, to converge Event Auditing Options abstract data. These versions of Windows do not implement Kerberos Policy Information abstract data.

Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later domain controllers use the Group Policy: Security Protocol Extension, as specified in [[MS-GPSB]](%5bMS-GPSB%5d.pdf#Section_6a07a06be62847659d910d63ba47fdc0) section 2.2.2 to converge Kerberos Policy Information abstract data and [MS-GPSB] section 2.2.4 to converge Event Auditing Options abstract data.

[<41> Section 3.1.1.2.1](#Appendix_A_Target_41): The following is a timeline of when each [**privilege**](#gt_d8092e10-b227-4b44-b015-511bb8178940) value was introduced. All privilege values continue to be supported in all subsequent versions of Windows according to the applicability lists at the beginning of this section.

| Name | Product |
| --- | --- |
| SE\_ASSIGNPRIMARYTOKEN\_NAME  "SeAssignPrimaryTokenPrivilege" | Windows NT 3.1 |
| SE\_AUDIT\_NAME  "SeAuditPrivilege" | Windows NT 3.1 |
| SE\_BACKUP\_NAME  "SeBackupPrivilege" | Windows NT 3.1 |
| SE\_CHANGE\_NOTIFY\_NAME  "SeChangeNotifyPrivilege" | Windows NT 3.1 |
| SE\_CREATE\_GLOBAL\_NAME  "SeCreateGlobalPrivilege" | Windows 2000 SP4, Windows XP operating system Service Pack 2 (SP2), and Windows Server 2003 |
| SE\_CREATE\_PAGEFILE\_NAME  "SeCreatePagefilePrivilege" | Windows NT 3.1 |
| SE\_CREATE\_PERMANENT\_NAME  "SeCreatePermanentPrivilege" | Windows NT 3.1 |
| SE\_CREATE\_TOKEN\_NAME  "SeCreateTokenPrivilege" | Windows NT 3.1 |
| SE\_DEBUG\_NAME  "SeDebugPrivilege" | Windows NT 3.1 |
| SE\_ENABLE\_DELEGATION\_NAME  "SeEnableDelegationPrivilege" | Windows 2000 |
| SE\_IMPERSONATE\_NAME  "SeImpersonatePrivilege" | Windows 2000 SP4, Windows XP SP2, and Windows Server 2003 |
| SE\_INC\_BASE\_PRIORITY\_NAME  "SeIncreaseBasePriorityPrivilege" | Windows NT 3.1 |
| SE\_INCREASE\_QUOTA\_NAME  "SeIncreaseQuotaPrivilege" | Windows NT 3.1 |
| SE\_LOAD\_DRIVER\_NAME  "SeLoadDriverPrivilege" | Windows NT 3.1 |
| SE\_LOCK\_MEMORY\_NAME  "SeLockMemoryPrivilege" | Windows NT 3.1 |
| SE\_MACHINE\_ACCOUNT\_NAME  "SeMachineAccountPrivilege" | Windows NT 3.5 |
| SE\_MANAGE\_VOLUME\_NAME  "SeManageVolumePrivilege" | Windows 2000 SP4 and Windows XP |
| SE\_PROF\_SINGLE\_PROCESS\_NAME  "SeProfileSingleProcessPrivilege" | Windows NT 3.1 |
| SE\_REMOTE\_SHUTDOWN\_NAME  "SeRemoteShutdownPrivilege" | Windows NT 3.1 |
| SE\_RESTORE\_NAME  "SeRestorePrivilege" | Windows NT 3.1 |
| SE\_SECURITY\_NAME  "SeSecurityPrivilege" | Windows NT 3.1 |
| SE\_SHUTDOWN\_NAME  "SeShutdownPrivilege" | Windows NT 3.1 |
| SE\_SYNC\_AGENT\_NAME  "SeSyncAgentPrivilege" | Windows 2000 |
| SE\_SYSTEM\_ENVIRONMENT\_NAME  "SeSystemEnvironment" | Windows NT 3.1 |
| SE\_SYSTEM\_PROFILE\_NAME  "SeSystemProfilePrivilege" | Windows NT 3.1 |
| SE\_SYSTEMTIME\_NAME  "SeSystemtimePrivilege" | Windows NT 3.1 |
| SE\_TAKE\_OWNERSHIP\_NAME  "SeTakeOwnershipPrivilege" | Windows NT 3.1 |
| SE\_TCB\_NAME  "SeTcbPrivilege" | Windows NT 3.1 |
| SE\_UNDOCK\_NAME  "SeUndockPrivilege" | Windows NT 3.1 |
| SE\_CREATE\_SYMBOLIC\_LINK\_NAME "SeCreateSymbolicLinkPrivilege" | Windows Vista and Windows Server 2008 |
| SE\_INC\_WORKING\_SET\_NAME "SeIncreaseWorkingSetPrivilege" | Windows Vista and Windows Server 2008 |
| SE\_RELABEL\_NAME  "SeRelabelPrivilege" | Windows Vista and Windows Server 2008 |
| SE\_TIME\_ZONE\_NAME "SeTimeZonePrivilege" | Windows Vista and Windows Server 2008 |
| SE\_TRUSTED\_CREDMAN\_ACCESS\_NAME "SeTrustedCredManAccessPrivilege" | Windows Vista and Windows Server 2008 |

[<42> Section 3.1.1.2.2](#Appendix_A_Target_42): Windows products implement the exact set of system access rights that the protocol supports for a given version. See the Windows behavior note in section 2.2.1.2 for a timeline of the system access introduction.

[<43> Section 3.1.1.3](#Appendix_A_Target_43): The default security descriptor that is assigned to newly created [**account objects**](#gt_b76eee27-064e-461b-81a9-fbf41e49928b) can be expressed in Security Description Definition Language (SDDL) as O:BAG:SYD:(A;;GA;;;BA)(A;;GX;;;WD).

See section 2.2.1.1.1 for the definitions of the generic access rights that are included in this security descriptor.

[<44> Section 3.1.1.3](#Appendix_A_Target_44): Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 domain controllers use the Netlogon Remote Protocol, as specified in [MS-NRPC] section 1.3.3.

Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later domain controllers use the Group Policy: Security Protocol Extension, as specified in [MS-GPSB] section 2.2.6.

[<45> Section 3.1.1.4](#Appendix_A_Target_45): The following is a timeline of when each secret name or name pattern was introduced. All secret names and name patterns continue to be available in subsequent versions of Windows according to the applicability lists at the beginning of this section.

| Secret name or name pattern | Product |
| --- | --- |
| Starts with "G$$" | Windows NT 3.1 |
| Starts with "G$" | Windows NT 3.1 |
| Starts with "L$" | Windows 2000 |
| Starts with "M$" | Windows 2000 |
| Starts with "\_sc\_" | Windows 2000 |
| Starts with "NL$" | Windows 2000 |
| Starts with "RasDialParams" | Windows 2000 |
| Starts with "RasCredentials" | Windows 2000 |
| Equal to "$MACHINE.ACC" | Windows NT 3.1 |
| Equal to "SAC" | Windows 2000 |
| Equal to "SAI" | Windows 2000 |
| Equal to "SANSC" | Windows 2000 |

The Trusted Domain Secret type is used only in Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0.

For replication of secrets, Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 use Netlogon-based replication, while Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later use [**Active Directory**](#gt_e467d927-17bf-49c9-98d1-96ddf61ddd90) replication.

[<46> Section 3.1.1.4](#Appendix_A_Target_46): By default, the security descriptor assigned to newly created [**secret objects**](#gt_9720ddb8-c802-40b7-8dba-5e7520c3396d) of type Local Secret can be expressed in Security Description Definition Language (SDDL) as O:BAG:SYD:(A;;GA;;;BA)(A;;GX;;;WD). This security descriptor implies that the secrets are shared between users by default, which means that a secret object created by an administrator is available to another administrator. An implementation can disallow this behavior by assigning a different security descriptor.

See section 2.2.1.1.1 for the definitions of the generic access rights that are included in this security descriptor.

[<47> Section 3.1.1.5](#Appendix_A_Target_47): The following is a timeline of when each information value was introduced. All information values continue to be available in subsequent versions of Windows according to the applicability lists at the beginning of this section.

| Name | Product |
| --- | --- |
| Name | Windows NT 3.1 |
| Flat Name | Windows 2000 |
| Security Identifier | Windows NT 3.1 |
| Trust Type | Windows 2000 |
| Trust Direction | Windows 2000 |
| Trust Attributes | Windows 2000 |
| Posix Offset | Windows NT 3.1 |
| Trust Incoming Passwords | Windows NT 3.51 |
| Trust Outgoing Passwords | Windows NT 3.51 |
| Forest Trust Information | Windows XP, Windows Server 2003 |
| Supported Encryption Types | Windows Vista, Windows Server 2008 |
| Security Descriptor | Windows NT 3.1 |

[<48> Section 3.1.1.6.1](#Appendix_A_Target_48): The default setting value is FALSE for Windows NT, Windows 2000, and Windows XP. The default setting value is TRUE for Windows Server 2003 and Windows Server 2003 R2 and later and for Windows Vista and later.

This setting can be set to FALSE on Windows Server 2003 and Windows Server 2003 R2 and later and on Windows Vista and later by setting a "non-0" value on the following REG\_DWORD registry value:

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\TurnOffAnonymousBlock

Changes made to this setting must take effect immediately.

Note that the Boolean meaning of the TurnOffAnonymousBlock registry value is reversed from that of the LsaRestrictAnonymous setting in section [3.1.1.6.1](#Section_3A635F0D5DB84EFEAF3D131B8B62AED5).

[<49> Section 3.1.4](#Appendix_A_Target_49): The Windows implementation of this protocol asks the RPC engine to perform a strict [**Network Data Representation (NDR)**](#gt_9ebf9540-2c31-43bc-bc56-4a932faabf2d) data consistency check at target level 5.0 (as specified in [MS-RPCE] section 3) in Windows 2000 Professional and later and in Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later.

[<50> Section 3.1.4](#Appendix_A_Target_50): The Windows implementation of this protocol asks the RPC engine to include support for both NDR and [**NDR64**](#gt_29385c51-3799-4da6-8291-32fc46a81970) transfer syntaxes, in addition to the negotiation mechanism for determining what transfer syntax will be used (as specified in [MS-RPCE] section 3) in Windows XP and later and in Windows Server 2003 and Windows Server 2003 R2 and later.

[<51> Section 3.1.4](#Appendix_A_Target_51): The Windows implementation of this protocol asks the RPC engine via the strict\_context\_handle attribute to reject use of context handles created by a method of a different RPC interface from this one, as specified in [MS-RPCE] section 3.

[<52> Section 3.1.4](#Appendix_A_Target_52): The following is a timeline of when each method was introduced. All methods continue to be available in subsequent versions of Windows according to the applicability list at the beginning of this section.

| Method | Product |
| --- | --- |
| [LsarClose (section 3.1.4.9.4)](#Section_99dd2d7ab0fc4c6d837a2b4d342383ae) | Windows NT 3.1 |
| [LsarEnumeratePrivileges (section 3.1.4.8.1)](#Section_e1c6e808de604dedb77d32f71a5a934a) | Windows NT 3.1 |
| [LsarQuerySecurityObject (section 3.1.4.9.1)](#Section_6b3291a21265498e8e9df7e28962255e) | Windows NT 3.1 |
| [LsarSetSecurityObject (section 3.1.4.9.2)](#Section_d4eb72865f194040a0c1d29136e0e58e) | Windows NT 3.1 |
| [LsarOpenPolicy (section 3.1.4.4.2)](#Section_2a482ccf1f8946938594855ff738ae8a) | Windows NT 3.1 |
| [LsarQueryInformationPolicy (section 3.1.4.4.4)](#Section_3564ba7084ea4f04a9dcdede9f96a8bf) | Windows NT 3.1 |
| [LsarSetInformationPolicy (section 3.1.4.4.6)](#Section_8a82ce8168e142da88a751096dcde022) | Windows NT 3.1 |
| [LsarCreateAccount (section 3.1.4.5.1)](#Section_841e32115be44b509f112d4941c40a30) | Windows NT 3.1 |
| [LsarEnumerateAccounts (section 3.1.4.5.2)](#Section_86f5e73b98c4423489cbd9ff5f327b73) | Windows NT 3.1 |
| [LsarCreateTrustedDomain (section 3.1.4.7.12)](#Section_373a4b1e1e8d45729c250bd7b045d3a3) | Windows NT 3.1 |
| [LsarEnumerateTrustedDomains (section 3.1.4.7.8)](#Section_3de62a51861e4373ae2ff3433cc10106) | Windows NT 3.1 |
| [LsarCreateSecret (section 3.1.4.6.1)](#Section_35a984a1d0024d60946db557ff4c46e0) | Windows NT 3.1 |
| [LsarOpenAccount (section 3.1.4.5.3)](#Section_355e2952abe447c396d9a2f4bd01cf3d) | Windows NT 3.1 |
| [LsarEnumeratePrivilegesAccount (section 3.1.4.5.4)](#Section_0e99240e58574b3d85c4d24f3155f6d4) | Windows NT 3.1 |
| [LsarAddPrivilegesToAccount (section 3.1.4.5.5)](#Section_8a542f26243d4341ada58fed194bfcf8) | Windows NT 3.1 |
| [LsarRemovePrivilegesFromAccount (section 3.1.4.5.6)](#Section_e92d5d073ded4d5299c63057312a37b3) | Windows NT 3.1 |
| [LsarGetSystemAccessAccount (section 3.1.4.5.7)](#Section_6d656257bfb9419fad578e775bf700cf) | Windows NT 3.1 |
| [LsarSetSystemAccessAccount (section 3.1.4.5.8)](#Section_3d73874ca2d7456aac220e7ece767a28) | Windows NT 3.1 |
| [LsarOpenTrustedDomain (section 3.1.4.7.1)](#Section_c59462aa9dca49e09ff1a5a009f0c64c) | Windows NT 3.1 |
| [LsarQueryInfoTrustedDomain (section 3.1.4.7.13)](#Section_e74460c7db0345c3ac3ca72a840e4943) | Windows NT 3.1 |
| [LsarSetInformationTrustedDomain (section 3.1.4.7.14)](#Section_9ea46cefcc724109ba1391eda6b713bc) | Windows NT 3.1 |
| [LsarOpenSecret (section 3.1.4.6.2)](#Section_8bf25269014f43fdb80f7a59a4883451) | Windows NT 3.1 |
| [LsarSetSecret (section 3.1.4.6.3)](#Section_21c1a153032c4869afc9186b2346dfab) | Windows NT 3.1 |
| [LsarQuerySecret (section 3.1.4.6.4)](#Section_e36cfffafd53437ea5a71a95cfdda4c1) | Windows NT 3.1 |
| [LsarLookupPrivilegeValue (section 3.1.4.8.2)](#Section_1f36a524be2640399047ad71899d0d8c) | Windows NT 3.1 |
| [LsarLookupPrivilegeName (section 3.1.4.8.3)](#Section_b4ae2755389741178a832b98874e3982) | Windows NT 3.1 |
| [LsarLookupPrivilegeDisplayName (section 3.1.4.8.4)](#Section_6ae3f7f030ed48c2a9bebe4a4617decd) | Windows NT 3.1 |
| [LsarDeleteObject (section 3.1.4.9.3)](#Section_8d0aa2dc22b64bc3b5d279b4b0ad7bce) | Windows NT 3.1 |
| [LsarEnumerateAccountsWithUserRight (section 3.1.4.5.9)](#Section_9c6ec6db534b41d5bbacfaa9ad31b380) | Windows NT 3.51 |
| [LsarEnumerateAccountRights (section 3.1.4.5.10)](#Section_920f27bbde33418bb6a9add38bcb4f57) | Windows NT 3.51 |
| [LsarAddAccountRights (section 3.1.4.5.11)](#Section_354b5a33670547f8adfbe291c633e9a8) | Windows NT 3.51 |
| [LsarRemoveAccountRights (section 3.1.4.5.12)](#Section_33613a1765c2403d83e9edbff71bc3de) | Windows NT 3.51 |
| [LsarQueryTrustedDomainInfo (section 3.1.4.7.2)](#Section_862cdc4f79e349738fd32e7836010eb5) | Windows NT 3.51 |
| [LsarSetTrustedDomainInfo (section 3.1.4.7.3)](#Section_d541b40a06ad4b3bbbfa5cf51a1f02d9) | Windows NT 3.51 |
| [LsarDeleteTrustedDomain (section 3.1.4.7.4)](#Section_633788c91e984555bc41d78a0de0f4a4) | Windows NT 3.51 |
| [LsarStorePrivateData (section 3.1.4.6.5)](#Section_b79c94fed7174ecf963ca200682921dc) | Windows NT 3.51 |
| [LsarRetrievePrivateData (section 3.1.4.6.6)](#Section_b46f3725d3de46b78245a14edeb278a1) | Windows NT 3.51 |
| [LsarOpenPolicy2 (section 3.1.4.4.1)](#Section_9456a9637c214710af77d0a2f5a72d6b) | Windows NT 3.51 |
| [LsarQueryInformationPolicy2 (section 3.1.4.4.3)](#Section_516f503c0230489db012e650b46b66a2) | Windows 2000 |
| [LsarSetInformationPolicy2 (section 3.1.4.4.5)](#Section_fc14e9aea26e4031809ea908dd3e13a3) | Windows 2000 |
| [LsarQueryTrustedDomainInfoByName (section 3.1.4.7.5)](#Section_5778eba18f9b4696b30434c58edbc5b1) | Windows 2000 |
| [LsarSetTrustedDomainInfoByName (section 3.1.4.7.6)](#Section_7a243aac9d5942eaba2754775c4f1573) | Windows 2000 |
| [LsarEnumerateTrustedDomainsEx (section 3.1.4.7.7)](#Section_14e37cf7b090497ca2e297e8425532a2) | Windows 2000 |
| [LsarCreateTrustedDomainEx (section 3.1.4.7.11)](#Section_6817095dd3414d0393d7e9bdca2d3eef) | Windows 2000 |
| [LsarQueryDomainInformationPolicy (section 3.1.4.4.7)](#Section_833ec261c4f649a4be211dd31a9525c0) | Windows 2000 |
| [LsarSetDomainInformationPolicy (section 3.1.4.4.8)](#Section_72900db3247b479c86f01c5b7727971a) | Windows 2000 |
| [LsarOpenTrustedDomainByName (section 3.1.4.7.9)](#Section_62ba989fbd1a459cb402bf315d44c6e1) | Windows 2000 |
| [LsarCreateTrustedDomainEx2 (section 3.1.4.7.10)](#Section_cc86a55db61948fd998a65cca15efeb9) | Windows 2000 |
| [LsarQueryForestTrustInformation (section 3.1.4.7.15)](#Section_fbc7a303997047a182cf0bd635a35bd1) | Windows XP, Windows Server 2003 |
| [LsarSetForestTrustInformation (section 3.1.4.7.16)](#Section_16be42bce85c4135b183aacb88106306) | Windows XP, Windows Server 2003 |

[<53> Section 3.1.4](#Appendix_A_Target_53): Some gaps in the [**opnum**](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7) numbering sequence correspond to opnums that are specified in [[MS-LSAT]](%5bMS-LSAT%5d.pdf#Section_1ba21e6fd8a9462c91534375f2020894). All other gaps in the opnum numbering sequence apply to Windows as follows.

| Opnum | Description |
| --- | --- |
| 1 | Used only locally by Windows, never remotely. |
| 5 | Not used by Windows. |
| 9 | Not used by Windows. |
| 21 | Not used by Windows. |
| 22 | Not used by Windows. |
| 52 | Not used by Windows. |
| 56 | Used only locally by Windows, never remotely. |
| 60 | Used only locally by Windows, never remotely. |
| 61 | Used only locally by Windows, never remotely. |
| 62 | Used only locally by Windows, never remotely. |
| 63 | Used only locally by Windows, never remotely. |
| 64 | Used only locally by Windows, never remotely. |
| 65 | Used only locally by Windows, never remotely. |
| 66 | Used only locally by Windows, never remotely. |
| 67 | Used only locally by Windows, never remotely. |
| 69 | Used only locally by Windows, never remotely. |
| 70 | Used only locally by Windows, never remotely. |
| 71 | Used only locally by Windows, never remotely. |
| 72 | Used only locally by Windows, never remotely. |
| 75 | Used only locally by Windows, never remotely. |

[<54> Section 3.1.4.4.1](#Appendix_A_Target_54): The Windows RPC server for this protocol ignores this parameter except for the **RootDirectory** field. It verifies whether the value is NULL and returns STATUS\_INVALID\_PARAMETER if it is not.

[<55> Section 3.1.4.4.2](#Appendix_A_Target_55): The Windows RPC server for this protocol ignores this parameter except for the **RootDirectory** field. It verifies whether the value is NULL and returns STATUS\_INVALID\_PARAMETER if it is not.

[<56> Section 3.1.4.4.3](#Appendix_A_Target_56): Windows XP and later, and Windows Server 2003 and Windows Server 2003 R2 and later return STATUS\_INVALID\_PARAMETER for this information class.

[<57> Section 3.1.4.4.3](#Appendix_A_Target_57): In the case of Windows 2000 Professional and later, and Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later, the Windows RPC server always throws an RPC\_NT\_PROCNUM\_OUT\_OF\_RANGE exception if the server is configured to emulate NT4 for PolicyDnsDomainInformation information level.

[<58> Section 3.1.4.4.5](#Appendix_A_Target_58): Windows XP and later, and Windows Server 2003 and Windows Server 2003 R2 and later return STATUS\_INVALID\_PARAMETER for this information class.

[<59> Section 3.1.4.4.5](#Appendix_A_Target_59): Windows 2000 Professional and later, and Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later behavior: The Windows RPC server always throws an RPC\_NT\_PROCNUM\_OUT\_OF\_RANGE exception if the server is configured to emulate NT4 for PolicyDnsDomainInformation information level.

[<60> Section 3.1.4.5.1](#Appendix_A_Target_60): Windows checks whether the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) is valid, but does not validate the structure of the SID.

[<61> Section 3.1.4.5.5](#Appendix_A_Target_61): Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2 ignore invalid [**LUIDs**](#gt_96b64af9-1896-4bde-b988-54d469c5affd) and return STATUS\_SUCCESS instead of STATUS\_INVALID\_PARAMETER.

[<62> Section 3.1.4.5.6](#Appendix_A_Target_62): Windows Vista and later do not allow removal of "SeAuditPrivilege", "SeChangeNotifyPrivilege", "SeImpersonatePrivilege", and "SeCreateGlobalPrivilege" from accounts represented with SIDs "S-1-5-19" and "S-1-5-20". Such requests are rejected with STATUS\_NOT\_SUPPORTED.

[<63> Section 3.1.4.5.9](#Appendix_A_Target_63): Furthermore, Windows checks that the caller is a member of Builtin Administrators.

[<64> Section 3.1.4.5.12](#Appendix_A_Target_64): Windows Vista and later and Windows Server 2008 and later do not allow removal of "SeAuditPrivilege", "SeChangeNotifyPrivilege", "SeImpersonatePrivilege", and "SeCreateGlobalPrivilege" from accounts represented with SIDs "S-1-5-19" and "S-1-5-20". Such requests are rejected with STATUS\_NOT\_SUPPORTED.

[<65> Section 3.1.4.6](#Appendix_A_Target_65): Windows 2000 Server, Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, and Windows Server 2008 support these methods. Windows 7 and later and Windows Server 2008 R2 and later support these methods by default, but can be configured not to support them.

[<66> Section 3.1.4.6.1](#Appendix_A_Target_66): Windows NT 4.0 and Windows 2000 Professional and later, and Windows NT 4.0, Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later limit the secret name length to 128 characters. Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2 return STATUS\_NAME\_TOO\_LONG for lengths that are greater than 128 characters. Windows Vista and later and Windows Server 2008 and later return STATUS\_INVALID\_PARAMETER for lengths that are greater than 128 characters.

[<67> Section 3.1.4.6.1](#Appendix_A_Target_67): Windows 2000 Professional and later, and Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later do not allow a secret whose name is prefixed by "G$$" to be created, and return STATUS\_INVALID\_PARAMETER to indicate this constraint failure to the caller.

[<68> Section 3.1.4.6.1](#Appendix_A_Target_68): Windows Server 2003 and Windows Server 2003 R2 and later, and Windows Vista and later do not allow the secret name to be "G$$", "G$", "L$", "M$", "\_sc\_", "NL$", "RasDialParams" or "RasCredentials". They return STATUS\_INVALID\_PARAMETER to indicate this constraint failure to the caller.

[<69> Section 3.1.4.6.1](#Appendix_A_Target_69): Global secrets (those that are prefixed with "G$") cannot be created on domain controllers on which the [**directory**](#gt_49ce3946-04d2-4cc9-9350-ebcd952b9ab9) service is stopped. A request to create a global secret on a domain controller on which the [**directory service**](#gt_c36db657-3138-4d9a-9289-ded5cbb8b40e) is stopped fails with status code STATUS\_DIRECTORY\_SERVICE\_REQUIRED.

[<70> Section 3.1.4.6.2](#Appendix_A_Target_70): Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later have a special case for secret name search for downlevel compatibility with Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51. If the secret name is in the form "G$$<NAME>", where "<NAME>" matches the name of a [**trusted domain**](#gt_f2f00d47-6cf2-4b32-b8f7-b63e38e2e9c4), the response is STATUS\_SUCCESS. In this case, secret information is Authentication Information of type TRUST\_AUTH\_TYPE\_CLEAR ([[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a) section 6.1.6.9.1.1, the **AuthType** field) from the [**trusted domain object**](#gt_f2ceef4e-999b-4276-84cd-2e2829de5fc4).

[<71> Section 3.1.4.6.3](#Appendix_A_Target_71): Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later have a special case for secret set operation for downlevel compatibility with Windows NT 3.1, Windows NT 3.5, and Windows NT 3.51. If the secret name is in the form "G$$<NAME>", where "<NAME>" matches the name of a trusted domain, the result is that the set request writes the secret value into the authentication information section of the trusted domain object. The access check in this case is identical to that required for setting authentication information on a trusted domain object, rather than that pertaining to changing a secret value.

[<72> Section 3.1.4.6.3](#Appendix_A_Target_72): If decryption of *EncryptedCurrentValue* fails, Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Server 2003 R2, and Windows Vista return STATUS\_UNKNOWN\_REVISION (0xC0000058); Windows Server 2008 and later and Windows 7 and later return STATUS\_INVALID\_PARAMETER\_1 (0xC00000EF).

[<73> Section 3.1.4.6.3](#Appendix_A_Target_73): If decryption of *EncryptedOldValue* fails, Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Server 2003 R2, and Windows Vista return STATUS\_UNKNOWN\_REVISION (0xC0000058); Windows Server 2008 and later and Windows 7 and later return STATUS\_INVALID\_PARAMETER\_1 (0xC00000EF).

[<74> Section 3.1.4.6.4](#Appendix_A_Target_74): Windows rejects the secret query requests of type "system" by returning STATUS\_ACCESS\_DENIED. Windows also rejects the secret query requests of type "local" from network clients with STATUS\_ACCESS\_DENIED.

[<75> Section 3.1.4.6.4](#Appendix_A_Target_75): If Windows 2000 Server, Windows Server 2003, or Windows Server 2003 R2 process a global secret with a value that has its **Length** field set to 0, they fill in the *EncryptedCurrentValue* with the following values before encryption.

1. Length = 0
2. MaximumLength = 0

Windows Server 2008 and later set the value of *EncryptedCurrentValue* to NULL.

[<76> Section 3.1.4.6.4](#Appendix_A_Target_76): If Windows 2000 Server, Windows Server 2003, or Windows Server 2003 R2 process a global secret with a value that has its **Length** field set to 0, they fill in the *EncryptedOldValue* with the following values before encryption.

1. Length = 0
2. MaximumLength = 0

Windows Server 2008 and later set the value of *EncryptedOldValue* to NULL.

[<77> Section 3.1.4.6.5](#Appendix_A_Target_77): If decryption of *EncryptedData* fails, Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Server 2003 R2, and Windows Vista return STATUS\_UNKNOWN\_REVISION (0xC0000058); Windows Server 2008 and later and Windows 7 and later return STATUS\_INVALID\_PARAMETER\_1 (0xC00000EF).

[<78> Section 3.1.4.7](#Appendix_A_Target_78): Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 use trusted domain objects on non–domain controllers to join a machine to a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). Therefore, trusted domain object methods are allowed on these products even when the machine is not a domain controller. There is, however, one extra check in this case, which is that the trusted domain object's security identifier has to be the same as the security identifier in Primary Domain Information. This also artificially limits the number of trusted domain objects on such systems to one.

[<79> Section 3.1.4.7.1](#Appendix_A_Target_79): Windows Server 2003 and Windows Server 2003 R2 and later disallow callers that do not have the AuthenticatedUsers SID in their token from accessing trusted domain objects. Requests by such users are rejected with STATUS\_ACCESS\_DENIED.

[<80> Section 3.1.4.7.1](#Appendix_A_Target_80): On Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later, Active Directory has to be running on the server in order for this request to succeed. Failing that, the STATUS\_DIRECTORY\_SERVICE\_REQUIRED status code is returned.

[<81> Section 3.1.4.7.3](#Appendix_A_Target_81): Read-only domain controllers are supported on servers running Windows Server 2008 and later. They return the STATUS\_OBJECT\_NAME\_NOT\_FOUND error.

[<82> Section 3.1.4.7.3](#Appendix_A_Target_82): Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 support these *InformationClass* values.

[<83> Section 3.1.4.7.4](#Appendix_A_Target_83): Read-only domain controllers are supported on servers running Windows Server 2008 and later. They return the STATUS\_OBJECT\_NAME\_NOT\_FOUND error.

[<84> Section 3.1.4.7.10](#Appendix_A_Target_84): Windows Server 2003 for Small Business Server 2003 does not support this message. Attempts to create a TDO in this environment causes the server to return STATUS\_NOT\_SUPPORTED\_ON\_SBS.

[<85> Section 3.1.4.7.10](#Appendix_A_Target_85): The operation is not supported on Windows Server 2003 for Small Business Server 2003.

[<86> Section 3.1.4.7.10](#Appendix_A_Target_86): Servers running Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, or Windows Server 2008 R2 return the STATUS\_INVALID\_DOMAIN\_STATE error when the TRUST\_ATTRIBUTE\_FOREST\_TRANSITIVE or the TRUST\_ATTRIBUTE\_CROSS\_ORGANIZATION bit is set in the **TrustAttributes** field of the *TrustedDomainInformation* input parameter.

[<87> Section 3.1.4.7.10](#Appendix_A_Target_87): Read-only domain controllers are supported on servers running Windows Server 2008 and later. They return the STATUS\_OBJECT\_NAME\_NOT\_FOUND error.

[<88> Section 3.1.4.7.11](#Appendix_A_Target_88): The operation is not supported on Windows Server 2003 for Small Business Server 2003.

[<89> Section 3.1.4.7.12](#Appendix_A_Target_89): The operation is not supported on Windows Server 2003 for Small Business Server 2003.

[<90> Section 3.1.4.7.13](#Appendix_A_Target_90): When not at DS\_BEHAVIOR\_WIN2003 [**forest functional level**](#gt_b3240417-ca43-4901-90ec-fde55b32b3b8), Windows Server 2003 and Windows Server 2003 R2 and later hide the presence of the TRUST\_ATTRIBUTE\_FOREST\_TRANSITIVE bit in the **Trust Attributes** field of a trusted domain object.

[<91> Section 3.1.4.7.14](#Appendix_A_Target_91): Servers running Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 return the STATUS\_INVALID\_INFO\_CLASS error when the information class is TrustedDomainInformationBasic.

[<92> Section 3.1.4.7.14](#Appendix_A_Target_92): Servers running Windows Server 2008 and later return the STATUS\_OBJECT\_NAME\_NOT\_FOUND error.

[<93> Section 3.1.4.9.1](#Appendix_A_Target_93): The server will not return the security descriptor of objects that it stores in Active Directory. It will return the security descriptor of objects in its local policy only. The objects stored in Active Directory include Global Secrets and trusted domain objects in Windows 2000 Server, Windows Server 2003, and Windows Server 2003 R2 and later. For objects that fall into this category, the server will return the STATUS\_NOT\_SUPPORTED status code.

[<94> Section 3.1.4.9.2](#Appendix_A_Target_94): The server will not return the security descriptor of objects that it stores in Active Directory. It will return the security descriptor of objects in its local policy only. The objects stored in Active Directory include Global Secrets and trusted domain objects. For objects that fall into this category, the server returns the STATUS\_NOT\_SUPPORTED status code.

[<95> Section 3.1.4.10](#Appendix_A_Target_95): On Windows Server 2008 and later, when processing the LsarOpenSecret (section 3.1.4.6.2) and LsarCreateSecret (section 3.1.4.6.1) methods, the length of the string is allowed to not be a multiple of 2. If **Length** is not a multiple of 2, the length of the Unicode string will be assumed to be **Length** – 1.

[<96> Section 3.1.4.10](#Appendix_A_Target_96): Windows NT, Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2 do not perform this check. On Windows Server 2008 and later, when processing the LsarOpenSecret and LSarCreateSecret methods, the **Buffer** field is allowed to contain zero or many NULL Unicode characters at the end of the string.

[<97> Section 3.1.4.10](#Appendix_A_Target_97): Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2 implementations of this protocol do not validate the Luid.HighPart field.

[<98> Section 3.1.4.10](#Appendix_A_Target_98): Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2 implementations of this protocol do not validate the Luid.LowPart field.

[<99> Section 3.1.4.10](#Appendix_A_Target_99): Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2 implementations of this protocol do not validate the Attributes field.

# Change Tracking

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

# Index

A

[Abstract data model](#section_0877fdc4184f40b0b378f50d6647d23e) 67

[server](#section_0877fdc4184f40b0b378f50d6647d23e) 67

Access

[checks](#section_e6003a157a4a46e6a763fe40522243ce) 84

[rights](#section_e6003a157a4a46e6a763fe40522243ce) 84

[Access Rights and Access Checks method](#section_e6003a157a4a46e6a763fe40522243ce) 84

[Account Object Methods method](#section_c6282c425da94b2c87d914e12e59e786) 98

Account objects

[data model](#section_d1a82c1b4d2b456fb4dfc70b461f3d42) 73

[example](#section_926b0d88b7c54fedbaea2a6d5b3f01c7) 161

[methods](#section_c6282c425da94b2c87d914e12e59e786) 98

[Accounts Rights data model](#section_3594f0d3574b4d55a0a435d23885f3ee) 69

[Applicability](#section_7c9812deb31d47a1a13fa91ae94fad37) 21

C

[Capability negotiation](#section_6469591cb3f444e298983ef6631a703c) 22

[Change tracking](#section_c88f39de37224368a70a6f23e1c72491) 216

[Closing handles](#section_bfa65c4ab83741cf819048671f803d4f) 88

[Closing Handles method](#section_bfa65c4ab83741cf819048671f803d4f) 88

[Common data types](#section_4d4678cf32154ecc8dc95a2aaa0e1eb0) 23

[Common object methods](#section_cf8100ee82d14b6d8cbca022ade325be) 151

[Common Object Methods method](#section_cf8100ee82d14b6d8cbca022ade325be) 151

D

[Data model - abstract](#section_0877fdc4184f40b0b378f50d6647d23e) 67

[server](#section_0877fdc4184f40b0b378f50d6647d23e) 67

[Data types](#section_4d4678cf32154ecc8dc95a2aaa0e1eb0) 23

[common - overview](#section_4d4678cf32154ecc8dc95a2aaa0e1eb0) 23

[Data validation](#section_ba5f1e31b2a742bd9a903650a6e5f6f5) 156

[Data Validation method](#section_ba5f1e31b2a742bd9a903650a6e5f6f5) 156

[Decryption - secret](#section_cce8ae7801384129954ec65e0c0bffed) 173

[DES-ECB-LM cipher definition](#section_32a2c8afdc6e4662918def333d570dd2) 175

[Directory service schema elements](#section_83412de34f814adb890f825c8e93ac3d) 66

E

[Elements - directory service schema](#section_83412de34f814adb890f825c8e93ac3d) 66

[Encryption - secret](#section_cce8ae7801384129954ec65e0c0bffed) 173

Events

[local - server](#section_a2404be78ec84d89acbd71aec5607edd) 160

[timer - server](#section_aeeb4da9f7b344c1b7c8b03085d88252) 160

[Examples](#section_3186df57d57c4dc380a77f3050a5ccca) 161

[manipulating account objects](#section_926b0d88b7c54fedbaea2a6d5b3f01c7) 161

[manipulating secret objects](#section_2ed414d6a6bf4743b6a41bc37a873321) 164

[manipulating trusted domain objects](#section_268547e310754857be88ab1d16ed8f1a) 167

[overview](#section_3186df57d57c4dc380a77f3050a5ccca) 161

[structure example of lsapr\_trusted\_domain\_auth\_blob](#section_047d8c3f466448d4883babdf7f11814c) 169

F

[Fields - vendor-extensible](#section_3b53470f421e4a939350da089b94c577) 22

[Full IDL](#section_d86ca799b1224fb6bfa05c99dd862b11) 178

G

[Glossary](#section_31ca2a310be44773bcef05ad6cd3ccfb) 8

H

Handles

[closing](#section_bfa65c4ab83741cf819048671f803d4f) 88

[obtaining](#section_879d309fe4d4424b98482913d14994aa) 84

I

[IDL](#section_d86ca799b1224fb6bfa05c99dd862b11) 178

[Implementer - security considerations](#section_ed292c30f14a4568b5992e69358144d8) 173

[Implementers - security considerations](#section_ed292c30f14a4568b5992e69358144d8) 173

[Index of security parameters](#section_150bab77381c48f1b5bb37f99d2b8dba) 177

[Informative references](#section_0b0becc93ca84a13acb7229746beb736) 14

[Initialization](#section_9d77e362923a4a719bfaff5437e5c38e) 79

[server](#section_9d77e362923a4a719bfaff5437e5c38e) 79

[Introduction](#section_a78325be25a1489ba8dae07ad9f9ea1a) 8

L

Local events

[LSAPR\_HANDLE\_rundown](#section_2184a151ce744e1e8ba725384f51dbf8) 160

[overview](#section_a2404be78ec84d89acbd71aec5607edd) 160

[server](#section_a2404be78ec84d89acbd71aec5607edd) 160

[LSA\_FOREST\_TRUST\_BINARY\_DATA structure](#section_d4859b44b7764237baa112dc28c19634) 63

[LSA\_FOREST\_TRUST\_COLLISION\_INFORMATION structure](#section_db0e91319a194cb5937a0e3a5767a0b2) 65

[LSA\_FOREST\_TRUST\_COLLISION\_RECORD structure](#section_32178d2cca744f538264af1906f95011) 64

[LSA\_FOREST\_TRUST\_COLLISION\_RECORD\_TYPE enumeration](#section_afc7d769a31748059f4585d5393b57af) 64

[LSA\_FOREST\_TRUST\_DOMAIN\_INFO structure](#section_451ac72fe9ad4a4f961fd04a2a5b1515) 63

[LSA\_FOREST\_TRUST\_INFORMATION structure](#section_2993ffabc0c846439a794ff7d31922dc) 64

[LSA\_FOREST\_TRUST\_RECORD structure](#section_08cf1a65ef7c46d3aa4d558f5135df3d) 61

[LSA\_FOREST\_TRUST\_RECORD\_TYPE enumeration](#section_700a91e8a1a44e1b9ad6096b3cf0bef0) 63

[LSAPR\_ACCOUNT\_ENUM\_BUFFER structure](#section_727c2d44879448969fba5e1725bc288e) 46

[LSAPR\_ACCOUNT\_INFORMATION structure](#section_98540c1c09cc4ee2934acdde3de0c77f) 46

[LSAPR\_ACL structure](#section_a9a03a855b084bb581c92c68751693ac) 35

[LSAPR\_AUTH\_INFORMATION structure](#section_cedb0d1bc7c0448099fc279b06f22a0c) 59

[LSAPR\_CR\_CIPHER\_VALUE structure](#section_782eda77b82e413487c9eb5e67f18f06) 48

[LSAPR\_LUID\_AND\_ATTRIBUTES structure](#section_03c834c0f3104e0c832eb6e7688364d1) 47

[LSAPR\_OBJECT\_ATTRIBUTES structure](#section_ad9e183d64744641a6d9d3796d2d604b) 33

[LSAPR\_POLICY\_ACCOUNT\_DOM\_INFO structure](#section_bfad54243e2043bd87f6d35b4253792e) 41

[LSAPR\_POLICY\_AUDIT\_EVENTS\_INFO structure](#section_d00fc364577d4ed0b3a5952d78b67695) 40

[LSAPR\_POLICY\_DNS\_DOMAIN\_INFO structure](#section_3e15a02e25d346aa9c608def03c824d2) 43

[LSAPR\_POLICY\_DOMAIN\_EFS\_INFO structure](#section_3ba6e751cf914d87a74c488bb927a54c) 44

[LSAPR\_POLICY\_PD\_ACCOUNT\_INFO structure](#section_b04175b3fedf4dda9034f754a10fe64e) 41

[LSAPR\_POLICY\_PRIMARY\_DOM\_INFO structure](#section_0f3f5d3f66d245a08c28ede86f4cd4a8) 40

[LSAPR\_POLICY\_PRIVILEGE\_DEF structure](#section_f36d47375b2f4bc08f29e7b4c71b7401) 65

[LSAPR\_POLICY\_REPLICA\_SRCE\_INFO structure](#section_fb7df2bb99e7402f833424d47e23ec00) 42

[LSAPR\_PRIVILEGE\_ENUM\_BUFFER structure](#section_c0278280b4b64538b3aaeb40f64f42fb) 65

[LSAPR\_PRIVILEGE\_SET structure](#section_a30a5720778442f4b03ab14f4e486bae) 47

[LSAPR\_SECURITY\_DESCRIPTOR structure](#section_d5cf869d674449cca67730ccb9217def) 35

[LSAPR\_SR\_SECURITY\_DESCRIPTOR structure](#section_5564065e3f3d4481a385367cc9b042c4) 34

[LSAPR\_TRUST\_INFORMATION structure](#section_71e86cddae194a029179a2a103b383a0) 48

[LSAPR\_TRUSTED\_CONTROLLERS\_INFO structure](#section_5382bd8969c646f2beb17b70e5befbc5) 51

[LSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB structure](#section_da8f32a10a164194810d06cc0698595a) 57

[LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION structure](#section_084fdb6b5bc349129aed0257159996dd) 55

[LSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL structure](#section_3b1c61fe6f074d83af543a381de5c5d1) 56

[LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION structure](#section_9f9feebce9e141c18c4802f83a227a14) 56

[LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL structure](#section_2e9e2c847b004fb18de588d4cfedd2b3) 56

[LSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2 structure](#section_e529d0945de44738adc4efa1a7d1106f) 57

[LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX structure](#section_f28f42b7173c4cda98093fe4a5213ab3) 52

[LSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2 structure](#section_dd92d4d9227f4ef1b42bef3f056f8aaa) 54

[LSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO structure](#section_71c5724f447e452c9cb9a0fd90d88594) 51

[LSAPR\_TRUSTED\_ENUM\_BUFFER structure](#section_78f8a2e44f3d40f5bdd19dacdf1c832c) 61

[LSAPR\_TRUSTED\_ENUM\_BUFFER\_EX structure](#section_b9b29ed6786e483f9e3b776eb014086b) 61

[LSAPR\_TRUSTED\_PASSWORD\_INFO structure](#section_33d7a9e4c9ca40219627337d89e656a3) 52

[LSAPR\_USER\_RIGHT\_SET structure](#section_dcaca8ef34a342dd85b698363eb108ff) 47

[LsarAddAccountRights method](#section_354b5a33670547f8adfbe291c633e9a8) 109

[LsarAddPrivilegesToAccount method](#section_8a542f26243d4341ada58fed194bfcf8) 104

[LsarClose method](#section_99dd2d7ab0fc4c6d837a2b4d342383ae) 155

[LsarCreateAccount method](#section_841e32115be44b509f112d4941c40a30) 99

[LsarCreateSecret method](#section_35a984a1d0024d60946db557ff4c46e0) 112

[LsarCreateTrustedDomain method](#section_373a4b1e1e8d45729c250bd7b045d3a3) 136

[LsarCreateTrustedDomainEx method](#section_6817095dd3414d0393d7e9bdca2d3eef) 134

[LsarCreateTrustedDomainEx2 method](#section_cc86a55db61948fd998a65cca15efeb9) 132

[LsarDeleteObject method](#section_8d0aa2dc22b64bc3b5d279b4b0ad7bce) 154

[LsarDeleteTrustedDomain method](#section_633788c91e984555bc41d78a0de0f4a4) 125

[LsarEnumerateAccountRights method](#section_920f27bbde33418bb6a9add38bcb4f57) 108

[LsarEnumerateAccounts method](#section_86f5e73b98c4423489cbd9ff5f327b73) 100

[LsarEnumerateAccountsWithUserRight method](#section_9c6ec6db534b41d5bbacfaa9ad31b380) 107

[LsarEnumeratePrivileges method](#section_e1c6e808de604dedb77d32f71a5a934a) 147

[LsarEnumeratePrivilegesAccount method](#section_0e99240e58574b3d85c4d24f3155f6d4) 103

[LsarEnumerateTrustedDomains method](#section_3de62a51861e4373ae2ff3433cc10106) 129

[LsarEnumerateTrustedDomainsEx method](#section_14e37cf7b090497ca2e297e8425532a2) 128

[LsarGetSystemAccessAccount method](#section_6d656257bfb9419fad578e775bf700cf) 105

[LsarLookupPrivilegeDisplayName method](#section_6ae3f7f030ed48c2a9bebe4a4617decd) 150

[LsarLookupPrivilegeName method](#section_b4ae2755389741178a832b98874e3982) 149

[LsarLookupPrivilegeValue method](#section_1f36a524be2640399047ad71899d0d8c) 148

[LsarOpenAccount method](#section_355e2952abe447c396d9a2f4bd01cf3d) 102

[LsarOpenPolicy method](#section_2a482ccf1f8946938594855ff738ae8a) 89

[LsarOpenPolicy2 method](#section_9456a9637c214710af77d0a2f5a72d6b) 88

[LsarOpenSecret method](#section_8bf25269014f43fdb80f7a59a4883451) 113

[LsarOpenTrustedDomain method](#section_c59462aa9dca49e09ff1a5a009f0c64c) 120

[LsarOpenTrustedDomainByName method](#section_62ba989fbd1a459cb402bf315d44c6e1) 131

[LsarQueryDomainInformationPolicy method](#section_833ec261c4f649a4be211dd31a9525c0) 96

[LsarQueryForestTrustInformation method](#section_fbc7a303997047a182cf0bd635a35bd1) 142

[LsarQueryInformationPolicy method](#section_3564ba7084ea4f04a9dcdede9f96a8bf) 92

[LsarQueryInformationPolicy2 method](#section_516f503c0230489db012e650b46b66a2) 90

[LsarQueryInfoTrustedDomain method](#section_e74460c7db0345c3ac3ca72a840e4943) 137

[LsarQuerySecret method](#section_e36cfffafd53437ea5a71a95cfdda4c1) 116

[LsarQuerySecurityObject method](#section_6b3291a21265498e8e9df7e28962255e) 151

[LsarQueryTrustedDomainInfo method](#section_862cdc4f79e349738fd32e7836010eb5) 121

[LsarQueryTrustedDomainInfoByName method](#section_5778eba18f9b4696b30434c58edbc5b1) 126

[LsarRemoveAccountRights method](#section_33613a1765c2403d83e9edbff71bc3de) 110

[LsarRemovePrivilegesFromAccount method](#section_e92d5d073ded4d5299c63057312a37b3) 105

[LsarRetrievePrivateData method](#section_b46f3725d3de46b78245a14edeb278a1) 118

[LsarSetDomainInformationPolicy method](#section_72900db3247b479c86f01c5b7727971a) 97

[LsarSetForestTrustInformation method](#section_16be42bce85c4135b183aacb88106306) 143

[LsarSetInformationPolicy method](#section_8a82ce8168e142da88a751096dcde022) 95

[LsarSetInformationPolicy2 method](#section_fc14e9aea26e4031809ea908dd3e13a3) 93

[LsarSetInformationTrustedDomain method](#section_9ea46cefcc724109ba1391eda6b713bc) 139

[LsarSetSecret method](#section_21c1a153032c4869afc9186b2346dfab) 115

[LsarSetSecurityObject method](#section_d4eb72865f194040a0c1d29136e0e58e) 153

[LsarSetSystemAccessAccount method](#section_3d73874ca2d7456aac220e7ece767a28) 106

[LsarSetTrustedDomainInfo method](#section_d541b40a06ad4b3bbbfa5cf51a1f02d9) 122

[LsarSetTrustedDomainInfoByName method](#section_7a243aac9d5942eaba2754775c4f1573) 127

[LsarStorePrivateData method](#section_b79c94fed7174ecf963ca200682921dc) 117

M

[Manipulating account objects example](#section_926b0d88b7c54fedbaea2a6d5b3f01c7) 161

[Manipulating secret objects example](#section_2ed414d6a6bf4743b6a41bc37a873321) 164

[Manipulating trusted domain objects example](#section_268547e310754857be88ab1d16ed8f1a) 167

[Message processing](#section_2c6f3cf9d7924e8b9af55470f636c20a) 79

[server](#section_2c6f3cf9d7924e8b9af55470f636c20a) 79

Messages

[common data types](#section_4d4678cf32154ecc8dc95a2aaa0e1eb0) 23

[overview](#section_f14230b5272f4966847f05dc833153ff) 23

[transport](#section_64ea7ac432ef44f6ab51ea2b5a1c2390) 23

Methods

[Access Rights and Access Checks](#section_e6003a157a4a46e6a763fe40522243ce) 84

[account object](#section_c6282c425da94b2c87d914e12e59e786) 98

[Account Object Methods](#section_c6282c425da94b2c87d914e12e59e786) 98

[Closing Handles](#section_bfa65c4ab83741cf819048671f803d4f) 88

[common object](#section_cf8100ee82d14b6d8cbca022ade325be) 151

[Common Object Methods](#section_cf8100ee82d14b6d8cbca022ade325be) 151

[Data Validation](#section_ba5f1e31b2a742bd9a903650a6e5f6f5) 156

[Obtaining Handles](#section_879d309fe4d4424b98482913d14994aa) 84

[policy object](#section_ab0d94f35f504e44beb69895a1d0f711) 88

[Policy Object Methods](#section_ab0d94f35f504e44beb69895a1d0f711) 88

[privilege](#section_1c99ae50d43341778fa656e173831686) 146

[Privilege Methods](#section_1c99ae50d43341778fa656e173831686) 146

[secret object](#section_c86d4a49e9dd43f88ab144f6baffa2a0) 111

[Secret Object Methods](#section_c86d4a49e9dd43f88ab144f6baffa2a0) 111

[trusted domain object](#section_3d7af62ec260489d8ab5c756e0b9fe57) 119

[Trusted Domain Object Methods](#section_3d7af62ec260489d8ab5c756e0b9fe57) 119

N

[Normative references](#section_6ee1b4aea37545a98fbc3cb2dc3f76cc) 13

O

[Obtaining handles](#section_879d309fe4d4424b98482913d14994aa) 84

[Obtaining Handles method](#section_879d309fe4d4424b98482913d14994aa) 84

[Overview (synopsis)](#section_d0d689888b6e4ea8a05029c3f28234e1) 14

P

[Parameters - security](#section_150bab77381c48f1b5bb37f99d2b8dba) 177

[Parameters - security index](#section_150bab77381c48f1b5bb37f99d2b8dba) 177

[PLSA\_FOREST\_TRUST\_BINARY\_DATA](#section_d4859b44b7764237baa112dc28c19634) 63

[PLSA\_FOREST\_TRUST\_COLLISION\_INFORMATION](#section_db0e91319a194cb5937a0e3a5767a0b2) 65

[PLSA\_FOREST\_TRUST\_COLLISION\_RECORD](#section_32178d2cca744f538264af1906f95011) 64

[PLSA\_FOREST\_TRUST\_DOMAIN\_INFO](#section_451ac72fe9ad4a4f961fd04a2a5b1515) 63

[PLSA\_FOREST\_TRUST\_INFORMATION](#section_2993ffabc0c846439a794ff7d31922dc) 64

[PLSA\_FOREST\_TRUST\_RECORD](#section_08cf1a65ef7c46d3aa4d558f5135df3d) 61

[PLSAPR\_ACCOUNT\_ENUM\_BUFFER](#section_727c2d44879448969fba5e1725bc288e) 46

[PLSAPR\_ACCOUNT\_INFORMATION](#section_98540c1c09cc4ee2934acdde3de0c77f) 46

[PLSAPR\_ACL](#section_a9a03a855b084bb581c92c68751693ac) 35

[PLSAPR\_AUTH\_INFORMATION](#section_cedb0d1bc7c0448099fc279b06f22a0c) 59

[PLSAPR\_CR\_CIPHER\_VALUE](#section_782eda77b82e413487c9eb5e67f18f06) 48

[PLSAPR\_LUID\_AND\_ATTRIBUTES](#section_03c834c0f3104e0c832eb6e7688364d1) 47

[PLSAPR\_OBJECT\_ATTRIBUTES](#section_ad9e183d64744641a6d9d3796d2d604b) 33

[PLSAPR\_POLICY\_ACCOUNT\_DOM\_INFO](#section_bfad54243e2043bd87f6d35b4253792e) 41

[PLSAPR\_POLICY\_AUDIT\_EVENTS\_INFO](#section_d00fc364577d4ed0b3a5952d78b67695) 40

[PLSAPR\_POLICY\_DNS\_DOMAIN\_INFO](#section_3e15a02e25d346aa9c608def03c824d2) 43

[PLSAPR\_POLICY\_DOMAIN\_EFS\_INFO](#section_3ba6e751cf914d87a74c488bb927a54c) 44

[PLSAPR\_POLICY\_PD\_ACCOUNT\_INFO](#section_b04175b3fedf4dda9034f754a10fe64e) 41

[PLSAPR\_POLICY\_PRIMARY\_DOM\_INFO](#section_0f3f5d3f66d245a08c28ede86f4cd4a8) 40

[PLSAPR\_POLICY\_PRIVILEGE\_DEF](#section_f36d47375b2f4bc08f29e7b4c71b7401) 65

[PLSAPR\_POLICY\_REPLICA\_SRCE\_INFO](#section_fb7df2bb99e7402f833424d47e23ec00) 42

[PLSAPR\_PRIVILEGE\_ENUM\_BUFFER](#section_c0278280b4b64538b3aaeb40f64f42fb) 65

[PLSAPR\_PRIVILEGE\_SET](#section_a30a5720778442f4b03ab14f4e486bae) 47

[PLSAPR\_SECURITY\_DESCRIPTOR](#section_d5cf869d674449cca67730ccb9217def) 35

[PLSAPR\_SR\_SECURITY\_DESCRIPTOR](#section_5564065e3f3d4481a385367cc9b042c4) 34

[PLSAPR\_TRUST\_INFORMATION](#section_71e86cddae194a029179a2a103b383a0) 48

[PLSAPR\_TRUSTED\_CONTROLLERS\_INFO](#section_5382bd8969c646f2beb17b70e5befbc5) 51

[PLSAPR\_TRUSTED\_DOMAIN\_AUTH\_BLOB](#section_da8f32a10a164194810d06cc0698595a) 57

[PLSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION](#section_084fdb6b5bc349129aed0257159996dd) 55

[PLSAPR\_TRUSTED\_DOMAIN\_AUTH\_INFORMATION\_INTERNAL](#section_3b1c61fe6f074d83af543a381de5c5d1) 56

[PLSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION](#section_9f9feebce9e141c18c4802f83a227a14) 56

[PLSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION\_INTERNAL](#section_2e9e2c847b004fb18de588d4cfedd2b3) 56

[PLSAPR\_TRUSTED\_DOMAIN\_FULL\_INFORMATION2](#section_e529d0945de44738adc4efa1a7d1106f) 57

[PLSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX](#section_f28f42b7173c4cda98093fe4a5213ab3) 52

[PLSAPR\_TRUSTED\_DOMAIN\_INFORMATION\_EX2](#section_dd92d4d9227f4ef1b42bef3f056f8aaa) 54

[PLSAPR\_TRUSTED\_DOMAIN\_NAME\_INFO](#section_71c5724f447e452c9cb9a0fd90d88594) 51

[PLSAPR\_TRUSTED\_ENUM\_BUFFER](#section_78f8a2e44f3d40f5bdd19dacdf1c832c) 61

[PLSAPR\_TRUSTED\_ENUM\_BUFFER\_EX](#section_b9b29ed6786e483f9e3b776eb014086b) 61

[PLSAPR\_TRUSTED\_PASSWORD\_INFO](#section_33d7a9e4c9ca40219627337d89e656a3) 52

[PLSAPR\_USER\_RIGHT\_SET](#section_dcaca8ef34a342dd85b698363eb108ff) 47

Policy object

[data model](#section_c939c70e8cf04d90a9cc1c5002951723) 67

[methods](#section_ab0d94f35f504e44beb69895a1d0f711) 88

[Policy Object Methods method](#section_ab0d94f35f504e44beb69895a1d0f711) 88

[POLICY\_AUDIT\_EVENT\_TYPE enumeration](#section_04036d49a4824b57995f11d539f2e5b7) 46

[POLICY\_AUDIT\_FULL\_QUERY\_INFO structure](#section_0ef0845ff20e4897ad2988c0c07be0f4) 43

[POLICY\_AUDIT\_FULL\_SET\_INFO structure](#section_3224400e3c404e64810a8b11341ba4c6) 42

[POLICY\_AUDIT\_LOG\_INFO structure](#section_3fff1c62e8b14bc8b18a3ba6458ec622) 39

[POLICY\_DOMAIN\_INFORMATION\_CLASS enumeration](#section_566a61fc2e9947c899ca62f7e22cb15d) 43

[POLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO structure](#section_afcc492012d348e0ab95a8989ebbd41d) 45

[POLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO structure](#section_38bd52a04514468fb342d7421a51a316) 44

[POLICY\_INFORMATION\_CLASS enumeration](#section_9ce0bb37fc6c4230b1097e1881660b83) 37

[POLICY\_LSA\_SERVER\_ROLE enumeration](#section_620010b4b4394d46893acb67246de5fc) 41

[POLICY\_LSA\_SERVER\_ROLE\_INFO structure](#section_d37dbc6504f34db8b40a4e9dd6c12520) 42

[POLICY\_MODIFICATION\_INFO structure](#section_c80ae9d5d0c14d5ca0ae77eae7bfac25) 42

[PPOLICY\_AUDIT\_FULL\_QUERY\_INFO](#section_0ef0845ff20e4897ad2988c0c07be0f4) 43

[PPOLICY\_AUDIT\_FULL\_SET\_INFO](#section_3224400e3c404e64810a8b11341ba4c6) 42

[PPOLICY\_AUDIT\_LOG\_INFO](#section_3fff1c62e8b14bc8b18a3ba6458ec622) 39

[PPOLICY\_DOMAIN\_KERBEROS\_TICKET\_INFO](#section_afcc492012d348e0ab95a8989ebbd41d) 45

[PPOLICY\_DOMAIN\_QUALITY\_OF\_SERVICE\_INFO](#section_38bd52a04514468fb342d7421a51a316) 44

[PPOLICY\_LSA\_SERVER\_ROLE\_INFO](#section_d37dbc6504f34db8b40a4e9dd6c12520) 42

[PPOLICY\_MODIFICATION\_INFO](#section_c80ae9d5d0c14d5ca0ae77eae7bfac25) 42

[Preconditions](#section_bfa60d10e5a141bfb310df06f4aff9ec) 21

[Prerequisites](#section_bfa60d10e5a141bfb310df06f4aff9ec) 21

Privilege

[data model](#section_1a92af76d45f42c3b67cf1dc61bd6ee1) 69

[methods](#section_1c99ae50d43341778fa656e173831686) 146

[Privilege Methods method](#section_1c99ae50d43341778fa656e173831686) 146

[Product behavior](#section_2a769a08e023459faebe4fb3f595c0b7) 194

Protocol Details

[overview](#section_8db483db276e425883eb2ed181af6aaa) 67

[PSECURITY\_QUALITY\_OF\_SERVICE](#section_0ddf315053b542a5b0ec518bce67738c) 37

[PSTRING](#section_94a41a4fbd5d4c3eafd4cc17e83a6e01) 34

[PTRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES](#section_7c519a643dc14be6a17d76817cff6e39) 60

[PTRUSTED\_POSIX\_OFFSET\_INFO](#section_b091ee7ef5c34b4885671b08ea002221) 52

R

[RC4 cipher usage](#section_1f5bd3edcfdd42aba2acf0786082bb21) 173

[References](#section_9538f24ec52342dd955393c94b7c6dd3) 13

[informative](#section_0b0becc93ca84a13acb7229746beb736) 14

[normative](#section_6ee1b4aea37545a98fbc3cb2dc3f76cc) 13

[Relationship to other protocols](#section_a3817e1e51794e76a7d026ac376ab6a2) 19

S

[Schema elements - directory service](#section_83412de34f814adb890f825c8e93ac3d) 66

Secret

[decryption](#section_cce8ae7801384129954ec65e0c0bffed) 173

[encryption](#section_cce8ae7801384129954ec65e0c0bffed) 173

[Secret Object Methods method](#section_c86d4a49e9dd43f88ab144f6baffa2a0) 111

Secret objects

[data model](#section_483f1b6e7b1443419ab29b99c01f896e) 73

[example](#section_2ed414d6a6bf4743b6a41bc37a873321) 164

[methods](#section_c86d4a49e9dd43f88ab144f6baffa2a0) 111

[Security](#section_06fb7531695d4f38886b4f795c01b7ab) 173

[implementer considerations](#section_ed292c30f14a4568b5992e69358144d8) 173

[parameter index](#section_150bab77381c48f1b5bb37f99d2b8dba) 177

[SECURITY\_IMPERSONATION\_LEVEL enumeration](#section_720cea10cee24c459084c6fa7d67d18d) 36

[SECURITY\_QUALITY\_OF\_SERVICE structure](#section_0ddf315053b542a5b0ec518bce67738c) 37

[Sequencing rules](#section_2c6f3cf9d7924e8b9af55470f636c20a) 79

[server](#section_2c6f3cf9d7924e8b9af55470f636c20a) 79

Server

[abstract data model](#section_0877fdc4184f40b0b378f50d6647d23e) 67

[Access Rights and Access Checks method](#section_e6003a157a4a46e6a763fe40522243ce) 84

[Account Object Methods method](#section_c6282c425da94b2c87d914e12e59e786) 98

[Closing Handles method](#section_bfa65c4ab83741cf819048671f803d4f) 88

[Common Object Methods method](#section_cf8100ee82d14b6d8cbca022ade325be) 151

[Data Validation method](#section_ba5f1e31b2a742bd9a903650a6e5f6f5) 156

[initialization](#section_9d77e362923a4a719bfaff5437e5c38e) 79

[local events](#section_a2404be78ec84d89acbd71aec5607edd) 160

[message processing](#section_2c6f3cf9d7924e8b9af55470f636c20a) 79

[Obtaining Handles method](#section_879d309fe4d4424b98482913d14994aa) 84

[Policy Object Methods method](#section_ab0d94f35f504e44beb69895a1d0f711) 88

[Privilege Methods method](#section_1c99ae50d43341778fa656e173831686) 146

[Secret Object Methods method](#section_c86d4a49e9dd43f88ab144f6baffa2a0) 111

[sequencing rules](#section_2c6f3cf9d7924e8b9af55470f636c20a) 79

[timer events](#section_aeeb4da9f7b344c1b7c8b03085d88252) 160

[timers](#section_f7e20aa204b34b32bec614a987a12ad2) 79

[Trusted Domain Object Methods method](#section_3d7af62ec260489d8ab5c756e0b9fe57) 119

[Server - overview](#section_63acb76d473a407dbc0c548118e53055) 67

[Standards assignments](#section_302bc4e169fb48cbabb20d817c8473e5) 22

[STRING structure](#section_94a41a4fbd5d4c3eafd4cc17e83a6e01) 34

[Structure example of lsapr\_trusted\_domain\_auth\_blob example](#section_047d8c3f466448d4883babdf7f11814c) 169

[System access rights data model](#section_dc5ec15888ec4dc09396e89e8b9a2fca) 72

T

[Timer events](#section_aeeb4da9f7b344c1b7c8b03085d88252) 160

[server](#section_aeeb4da9f7b344c1b7c8b03085d88252) 160

[Timers](#section_f7e20aa204b34b32bec614a987a12ad2) 79

[server](#section_f7e20aa204b34b32bec614a987a12ad2) 79

[Tracking changes](#section_c88f39de37224368a70a6f23e1c72491) 216

[Transport](#section_64ea7ac432ef44f6ab51ea2b5a1c2390) 23

[Transport - message](#section_64ea7ac432ef44f6ab51ea2b5a1c2390) 23

[Trusted Domain Object Methods method](#section_3d7af62ec260489d8ab5c756e0b9fe57) 119

Trusted domain objects

[data model](#section_0228f75e9725479cb4cd1cacd667343c) 75

[example](#section_268547e310754857be88ab1d16ed8f1a) 167

[methods](#section_3d7af62ec260489d8ab5c756e0b9fe57) 119

[TRUSTED\_DOMAIN\_SUPPORTED\_ENCRYPTION\_TYPES structure](#section_7c519a643dc14be6a17d76817cff6e39) 60

[TRUSTED\_INFORMATION\_CLASS enumeration](#section_360691136c3845e8920e17f8ef36f578) 49

[TRUSTED\_POSIX\_OFFSET\_INFO structure](#section_b091ee7ef5c34b4885671b08ea002221) 52

V

[Validation - data](#section_ba5f1e31b2a742bd9a903650a6e5f6f5) 156

[Vendor-extensible fields](#section_3b53470f421e4a939350da089b94c577) 22

[Versioning](#section_6469591cb3f444e298983ef6631a703c) 22