[MS-DISO]:
Domain Interactions System Overview

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release (beta) version of this technology. This Open Specification is final

Copyright © 2011 Microsoft Corporation.

documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release (beta) versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

This document provides an overview of the Domain Interactions System Overview Protocol Family. It is intended for use in conjunction with the Microsoft Protocol Technical Documents, publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts. It assumes that the reader is either familiar with the aforementioned material or has immediate access to it.

A Protocol Family System Document does not require the use of Microsoft programming tools or programming environments in order to implement the Protocols in the System. Developers who have access to Microsoft programming tools and environments are free to take advantage of them.

Abstract

Microsoft Windows networks are often configured with a domain controller providing centralized storage of accounts and administration of many computers. Many network related operations depend on domains in order to complete various tasks. The Domain Interactions System includes the most common domain interaction tasks such as locating a domain controller, joining a domain, and removing a domain member.This document specifies how the protocols that comprise the offerings from Microsoft are used together to maintain a relationship with the domain. This includes protocols that are used to communicate with a domain controller and maintain state, protocols that are used to augment authentication and authorization actions, and protocols that are used to interact with domain controllers.

Revision Summary

Date	Revision History	Revision Class	Comments
11/06/2009	0.1	Major	First Release.
12/18/2009	1.0	Major	Updated and revised the technical content.
01/29/2010	2.0	Major	Updated and revised the technical content.
03/12/2010	3.0	Major	Updated and revised the technical content.
04/23/2010	4.0	Major	Updated and revised the technical content.
06/04/2010	5.0	Major	Updated and revised the technical content.
07/16/2010	6.0	Major	Significantly changed the technical content.
08/27/2010	7.0	Major	Significantly changed the technical content.
10/08/2010	8.0	Major	Significantly changed the technical content.
11/19/2010	9.0	Major	Significantly changed the technical content.

Copyright © 2011 Microsoft Corporation.

Date	Revision History	Revision Class	Comments
01/07/2011	10.0	Major	Significantly changed the technical content.
02/11/2011	11.0	Major	Significantly changed the technical content.
03/25/2011	12.0	Major	Significantly changed the technical content.
05/06/2011	13.0	Major	Significantly changed the technical content.
06/17/2011	13.1	Minor	Clarified the meaning of the technical content.

Copyright © 2011 Microsoft Corporation.

Contents

Copyright © 2011 Microsoft Corporation.

Copyright © 2011 Microsoft Corporation.

Copyright © 2011 Microsoft Corporation.

Copyright © 2011 Microsoft Corporation.

Copyright © 2011 Microsoft Corporation.

Copyright © 2011 Microsoft Corporation.

1 Introduction

A "Defined Task" is a logical procedure that uses one or more protocols or systems to accomplish a specific goal. This Defined Task System Document describes a set of Tasks that are centered on domain interactions and management.

In conjunction with Protocol Technical Documents (TDs), this Defined Task System Document describes the rules for information exchange relevant to the Tasks and the protocols that are used to operate or communicate with Windows client operating systems and selected Windows Server scenarios (those included in published TDs) in their various environments.

Microsoft Windows networks are often configured with a domain controller providing centralized services such as storage of accounts and administration of many computers (domain clients). This document describes how specific protocols are used together to maintain a relationship with the domain when domain clients interact with a domain.

This document is organized as follows:

- Section 1, "Introduction", describes what is covered in this document, provides a list of terms defined in this document, as well as terms used in this document but defined elsewhere in the documentation set, and provides a list of references that apply to the overall Windows Protocols System.
- Section 2, "Overview", introduces the Windows Protocols System. It provides a high-level map of how the systems and individual protocols in MCPP relate to each other.
- Section 3, "Background Knowledge and System-Specific Concepts", presents the background information that the reader would reasonably need to know to understand and implement the diverse set of protocols that exist in the MCPP document set. It includes concepts related to domains interaction such as types of domains, accounts, and services.

The following sections cover the defined tasks in this document:

- Section 5, "Locating a Domain Controller", helps a client to locate a domain controller using the Domain Name System (DNS) infrastructure, NetBIOS infrastructure, or both, available to the client.
- Section 6, "Joining a Domain Using Predefined Account", presents the process for a client to join a domain when the client already has an account on the domain.
- Section 7, "Joining a Domain by Creating an Account via SAM", presents the process for a client to join a domain by creating a new account using the MS-SAMR protocol.
- Section 8, "Joining a Domain by Creating an Account via LDAP", presents the process for a client to join a domain by creating a new account using the LDAP protocol.
- Section 9, "Removing a Domain Member", removes a client computer from the domain.

The tasks in sections 6 through 9 all depend on the local [MS-NRPC] server for locating a domain controller. Of the three domain join tasks, the Joining a Domain with a Predefined Account task establishes a base level to understand the sequence of operations. The Joining a Domain by Creating an Account via SAM task allows people writing domain controllers to know how the client will behave when joining the domain. The Joining a Domain by Creating an Account via LDAP task lets people writing domain controllers see how joining the domain can be done without the full implementation on the client side of protocols such as SAM, RPC, and SMB.

Copyright © 2011 Microsoft Corporation.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Active Directory
Authenticate
Binding
Credential
Directory
Distinguished Name (DN)
Domain
Domain Account
Domain Controller (DC)
Domain Member (member machine)
Domain Naming Context (domain NC)
Domain Name System (DNS)
Fully Qualified Domain Name (FQDN) (1) (2)
Group
Lightweight Directory Access Protocol (LDAP)
Naming Context (NC)
NetBIOS
Netlogon
Policy
Read-Only Domain Controller (RODC)
Security Account Manager (SAM)
Security Identifier (SID)
Server Message Block (SMB)
Service Resource Record (SRV)
Trusted Domain

The following terms are defined in [MS-ADTS]:

Active Directory Domain Services (AD DS)
Active Directory Lightweight Directory Services (AD LDS)
Filtered Attribute Set
forest

The following terms are defined in [MS-NRPC]:

Shared Secret

The following terms are specific to this document:

Account: A synonym for **security principal** or **principal**.

Account Database: The portion of the directory that maintains the accounts for the **principals** of the domain. In **Windows NT-4 style domains**, the account database includes all information in the NT domain; in AD-style domains, the account database contains a subset of the entire LDAP-accessible directory the AD-style domain hosts.

AD-Style Domain: A domain comprising Windows 2000 Server, Windows Server 2003, Windows Server 2003 R2, Windows Server 2008, or Windows Server 2008 R2 server computers. AD-style domains implement Active Directory (AD), LDAP, Kerberos authentication, and advanced configurations and features not supported in NT 4-style domains.

Client: Synonym for client computer.

Copyright © 2011 Microsoft Corporation.

Client Computer: A computer that is not a **domain controller server**; the computer may or may not be joined to a domain.

directory string: A string syntax specified in [RFC2252] section 6.10.

Domain Client: A client computer that is joined to a domain. The domain client can be a client or a server that offers other services to its clients. When the domain client acts as a supplicant to another domain client, the supplicant is referred to as a **domain client in a workstation role** and the latter as a domain client in a server role.

Domain Client in a Client Role: A domain member that acts as a supplicant to another domain client.

Domain Client in a Workstation Role: A domain member that offers other services to other domain clients.

Domain Controller Server: A domain member, which can be a client or a server that offers other services to its clients. When the domain client acts as a supplicant to another domain client, the supplicant is referred to as a **domain client in a workstation role** and the latter as a domain client in a server role.

Identity: An account that represents a person (user account), an application (service account), and computers that participate in the domain (machine accounts). A password is used by the system as proof of an identity.

Member Server: A server that is joined to a domain and is not a domain controller. Member servers typically function as file servers, application servers, and so on and defer user authentication to the domain controller.

Predefined Account: A machine account created in the directory by a domain administrator before a machine is associated with the account during domain join (see section 6).

Principal: A synonym of **security principal**.

Server: A domain controller. Used as a synonym for domain controller in this document.

Single Sign-On: A process that enables a user with a domain account to log on to a network once and gain access to all network resources.

Security Principal: An entity associated with a human user or a program that can be authenticated. At a minimum, it has two basic attributes, a name and an identifier, that uniquely identifies it and makes it meaningful to the system, administrators, and users. A **security principal** is also known as a **principal** or an account.

Trusted Third Party: A trusted third party issues signed statements to stated parties enabling those stated parties to act on another identity's behalf for a certain amount of time. It is trusted to perform a set of specialized functions, such as a security token service that provides authentication and **single sign-on** services to Web services. ([MSDN-SUBSYSDSGN]). As a trusted authentication service on the network, this service knows all passwords and can grant access to any server. It is convenient, but also the single point of failure and requires a high level of physical security. For the Kerberos authentication protocol, the trusted third party arbitrator is a server known as a Key Distribution Center (KDC) which runs the Kerberos daemons.

Windows NT-4 Style Domain: A domain comprised of Windows NT 4.0 servers with an account database that includes all the information in the domain. Windows NT 4.0 style domains do not implement Active Directory (AD), LDAP directories or Kerberos authentication.

Copyright © 2011 Microsoft Corporation.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. Note that in [RFC2119] terms, most of these specifications should be imperative, to ensure interoperability. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

Any specification that does not explicitly use one of these terms is mandatory, exactly as if it used MUST.

1.2 References

References to Microsoft Open Specification documents do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[FIPS186-2] U.S. Department of Commerce/National Institute of Standards and Technology, "Federal Information Processing Standards Publication 186-2: Digital Signature Standard (DSS)", January 2000, http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf

[MS-ADSC] Microsoft Corporation, "Active Directory Schema Classes".

[MS-ADSO] Microsoft Corporation, "Active Directory System Overview".

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

[MS-APDS] Microsoft Corporation, "Authentication Protocol Domain Support Specification".

[MS-AUTHSO] Microsoft Corporation, "Windows Authentication Services System Overview".

[MS-CAESO] Microsoft Corporation, "Certificate Autoenrollment System Overview".

[MS-CASO] Microsoft Corporation, "Certification Authority System Overview".

[MS-CIFS] Microsoft Corporation, "Common Internet File System (CIFS) Protocol Specification".

[MS-DFSC] Microsoft Corporation, "Distributed File System (DFS): Referral Protocol Specification".

[MS-DPSP] Microsoft Corporation, "Digest Protocol Extensions".

[MS-DRDM] Microsoft Corporation, "Directory Replication and Data Management (DRDM) Remote Protocol Specification".

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

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

[MS-FSSO] Microsoft Corporation, "File Access Services System Overview".

[MS-GPSO] Microsoft Corporation, "Group Policy System Overview".

[MS-KILE] Microsoft Corporation, "Kerberos Protocol Extensions".

Copyright © 2011 Microsoft Corporation.

[MS-LSAD] Microsoft Corporation, "Local Security Authority (Domain Policy) Remote Protocol Specification".

[MS-MAIL] Microsoft Corporation, "Remote Mailslot Protocol Specification".

[MS-MQSO] Microsoft Corporation, "Message Queuing System Overview".

[MS-NAPSO] Microsoft Corporation, "Network Access Protection System Overview".

[MS-NBTE] Microsoft Corporation, "NetBIOS over TCP (NetBT) Extensions".

[MS-NLMP] Microsoft Corporation, "NT LAN Manager (NTLM) Authentication Protocol Specification".

[MS-NRPC] Microsoft Corporation, "Netlogon Remote Protocol Specification".

[MS-PAC] Microsoft Corporation, "Privilege Attribute Certificate Data Structure".

[MS-PSSO] Microsoft Corporation, "Print Services System Overview".

[MS-RCMP] Microsoft Corporation, "Remote Certificate Mapping Protocol Specification".

[MS-RMSO] Microsoft Corporation, "Rights Management Services System Overview".

[MS-SAMR] Microsoft Corporation, "Security Account Manager (SAM) Remote Protocol Specification (Client-to-Server)".

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

[MS-SMB2] Microsoft Corporation, "Server Message Block (SMB) Version 2 Protocol Specification".

[MS-SNTP] Microsoft Corporation, "Network Time Protocol (NTP) Authentication Extensions".

[MS-SPNG] Microsoft Corporation, "Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) Extension".

[MS-TPSO] Microsoft Corporation, "Transaction Processing Services System Overview".

[MS-WMSO] Microsoft Corporation, "Windows Management Services System Overview".

[MS-WSO] Microsoft Corporation, "Windows System Overview".

[MS-WSUSO] Microsoft Corporation, "Windows Server Update Services System Overview".

[RFC1001] Network Working Group, "Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods", STD 19, RFC 1001, March 1987, http://www.ietf.org/rfc/rfc1001.txt

[RFC1002] Network Working Group, "Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications", STD 19, RFC 1002, March 1987, http://www.ietf.org/rfc/rfc1002.txt

[RFC1034] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987, http://www.ietf.org/rfc/rfc1034.txt

[RFC1035] Mockapetris, P., "Domain Names - Implementation and Specification", STD 13, RFC 1035, November 1987, http://www.ietf.org/rfc/rfc1035.txt

[RFC1769] Mills, D., "Simple Network Time Protocol (SNTP)", RFC 1769, March 1995, http://www.ietf.org/rfc/rfc1769.txt

Copyright © 2011 Microsoft Corporation.

[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

[RFC2247] Kille, S., Wahl, M., Grimstad, A., and Huber, R., "Using Domains in LDAP/X.500 Distinguished Names", RFC 2247, January 1998, http://www.ietf.org/rfc/rfc2247.txt

[RFC2251] Wahl, M., Howes, T., and Kille, S., "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997, http://www.ietf.org/rfc/rfc2251.txt

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

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., et al., "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999, http://www.ietf.org/rfc/rfc2617.txt

[RFC2782] Gulbrandsen, A., Vixie, P., and Esibov, L., "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000, http://www.ietf.org/rfc/rfc2782.txt

[RFC2831] Leach, P., and Newman, C., "Using Digest Authentication as a SASL Mechanism", RFC 2831, May 2000, http://www.ietf.org/rfc/rfc2831.txt

[RFC3596] Thomson, S., Huitema, C., Ksinant, V., and Souissi, M., "DNS Extensions to Support IP version 6", RFC 3596, October 2003, http://www.ietf.org/rfc/rfc3596.txt

[RFC3629] Yergeau, F., "UTF-8, A Transformation Format of ISO 10646", STD 63, RFC 3629, November 2003, http://www.ietf.org/rfc/rfc3629.txt

[RFC4120] Neuman, C., Yu, T., Hartman, S., and Raeburn, K., "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005, http://www.ietf.org/rfc/rfc4120.txt

1.2.2 Informative References

[MS-ADMS] Microsoft Corporation, "Shared Abstract Data Model Elements".

[MS-BRWS] Microsoft Corporation, "Common Internet File System (CIFS) Browser Protocol Specification".

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary".

[MSDN-SUBSYSDSGN] Microsoft Corporation, "Trusted Subsystem Design", http://msdn.microsoft.com/en-us/library/aa905320.aspx

[RFC819] Su, Z.S., and Postel, J., "The Domain Naming Convention for Internet User Applications", RFC 819, August 1982, http://www.ietf.org/rfc/rfc0819.txt

[RFC1305] Mills, D. L., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, March 1992, http://www.ietf.org/rfc/rfc1305.pdf

[RFC2052] Gulbrandsen, A., and Vixie, P., "A DNS RR for specifying the location of services (DNS SRV)", RFC 2052, October 1996, http://www.ietf.org/rfc/rfc2052.txt

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000, http://www.ietf.org/rfc/rfc2743.txt

[RFC4086] Eastlake III, D., Schiller, J., and Crokcer, S., "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005, http://www.ietf.org/rfc/rfc4086.txt

Copyright © 2011 Microsoft Corporation.

[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

Copyright © 2011 Microsoft Corporation.

2 Overview

Section 1, "Introduction" primarily describes this Defined Task System Document and introduces the Tasks being documented.

2.1 Summary

Microsoft Windows networks are often configured with a domain controller providing centralized storage of accounts and administration of many machines. Many network related operations depend on domains in order to complete various tasks. The Domain Interactions System describes some of these tasks, including:

- Locating a domain controller using DNS and NetBIOS.
- Joining a domain using a predefined account.
- Joining a domain by creating an account via the Security Account Manager (MS-SAMR) RPC protocol.
- Joining a domain by creating an account via the Lightweight Directory Access Protocol (LDAP) protocol.
- Removing a domain member.

The Domain Interactions System includes protocols that are used to communicate with a domain controller and maintain state. It also includes protocols that are used to augment authentication and authorization actions, and protocols that are used to interact with domain controllers. The relationships among the protocols that implement that functionality are complicated, and this document provides the framework necessary to understand them.

The domain controller serves a central role in an enterprise network by functioning as the root of authority for sets of users and computers. A domain controller aggregates functionality relating to identity management, authentication, authorization, and other management policy. Clients of the domain functionality in turn rely upon the domain controller to establish secure communication, authorize requests, and apply policy. A client of the domain may itself be a server of some other role, for example, a file server that is handling the file storage needs of other client workstations.

2.2 List of Tasks

The following tasks are described in this system:

Locating a Domain Controller: This set of tasks describes locating domain controllers using DNS, NetBIOS, or both.

Joining a Domain Using a Predefined Account: This task describes how a computer joins a domain using an account that is already configured.

Joining a Domain by Creating an Account via SAM: This task describes how a computer joins a domain after creating a new account via the Security Account Manager (MS-SAMR) RPC protocol.

Joining a Domain by Creating an Account via LDAP: This task describes how a computer joins a domain after creating a new account via the Lightweight Directory Access Protocol (LDAP) protocol.

Removing a Domain Member: This task describes how a member is removed from the domain.

Copyright © 2011 Microsoft Corporation.

2.3 Relevant Standards

The Domain Interactions System uses and extends the following standards:

Lightweight Directory Access Protocol (LDAP), as specified in [RFC2247]

Kerberos Authentication Protocol, as specified in [RFC4120]. This standard is used for authentication.

Simple Network Time Protocol (SNTP), as specified in [RFC1769]. This standard is an adaptation of the Network Time Protocol (NTP) used to synchronize computer clocks in the Internet. SNTP can be used when the ultimate performance of the full NTP implementation described in [RFC1305] is not needed or justified.

Domain Names - Concepts and Facilities, as specified in [RFC1034]. This standard is used for DNS and domain naming concepts.

Domain Names - Implementation and Specification, as specified in [RFC1035]. This standard is used for DNS and provides details of the domain system and protocol.

DNS Extensions to Support IP Version 6, as specified in [RFC3596]. This standard is used for DNS to support hosts running IP version 6 (IPv6).

UTF-8, A Transformation Format of ISO 10646, as specified in [RFC3629]. This standard is used for string data type specification.

Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods, as specified in [RFC1001]. This standard is used for NetBIOS services.

Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications, as specified in [RFC1002]. This standard is used for NetBIOS services.

Copyright © 2011 Microsoft Corporation.

3 Background Knowledge and System-Specific Concepts

This section identifies the theoretical and practical information needed to understand this document and the Tasks in this system, and summarizes:

- Background knowledge that is required to understand this document.
- Concepts that are specific to the Tasks in this system.

3.1 Domains

The purpose of the **domain** is to provide a centrally managed repository for accounts on a network. Entities within the domain can **authenticate** each other as part of other tasks, and can leverage the domain for authorization information to determine access to network resources. The domain system evolved from a time when each **server** on a network had its own **account database** and management. This required far more management on each server over time, and increased management for the users, who often had multiple accounts spread over different servers. Domains are directly analogous to Kerberos *realms*, as introduced in [RFC4120] section 1.2.

Specifically, a domain provides the following services:

- A common set of identifiers that can be centrally managed.
- Mechanisms for authenticating actors within the system to one another, based on those identifiers.
- Mechanisms for associating additional authorization information with those identifiers, allowing more efficient management of access control.
- Mechanisms for managing the systems and associated operating environments

In practice, domains are collections of users and computers, with the **domain controllers (DC)** serving the dual roles of hosting all the information about the domain, and serving as the **Trusted Third Party** for the domain. Typically, users use the domain to authenticate to servers, although any **principal** in the domain can make use of this functionality.

Domains are a step toward **single sign-on**, a concept by which a user authenticates once, and then can access all the relevant servers and other resources he or she needs. Rather than managing multiple sets of credentials, a single user **identity** and password (or equivalent) can be maintained by the user, and is applicable to all the tasks that a user needs to perform.

Originally, servers and domains were more concerned with correctly identifying a user making a request, to ensure that protection and authorization policies were enforced correctly. As threats became more sophisticated, it became equally necessary that the **clients** were able to authenticate the servers as well.

Also evolving at this time was the aggregation of servers into central management within corporate networks. While many servers originally were installed to support relatively small workgroups, the consolidation of identities also drove a desire for corporate IT managers to enforce company policy over more aspects of the client machines. Since all identities were in the domain, it was a natural place to root the policy system, allowing the IT managers to create and distribute policy that controlled the client machines.

To meet these needs and evolve to mitigate new threats, the general requirements of a domain are:

1. Designation of certain servers as DC that host the full account set.

Copyright © 2011 Microsoft Corporation.

2. Designation of the rest of the servers as **member servers** that defer authentication of users to the DCs.

3. A relatively simple authentication protocol (NTLM) that can authenticate a client to a server.

4. Supplanting this protocol with a protocol that provides mutual authentication of client and server (Kerberos).

5. Providing these authentication protocols to all members of the domain for use between clients and servers.

6. Providing these authentication protocols between client machines and DCs for the purpose of authenticating users as they begin using their machines interactively.

7. Enforce policies based on the membership of these users and machines in the domain (domain policy).

8. Further refine the policies based on expressing per-customer business need as part of the domain structure (group policy).

   These satisfied the general requirements of the domain. Further functionality has been included:

9. Support for additional authentication schemes, such as Digest and SSL/TLS.

Domains are typically implemented with multiple computers acting as DCs. This allows redundancy in deployment and guards against disruptive loss of service. In general, a client of the domain services treats all DCs as equivalent. The domain typically enforces loose consistency, guaranteeing only that all replication will eventually be put into an inactive state. As such, the client of the domain MUST treat the DC that it has selected as authoritative for its domain (subject to SID Filtering; see [MS-PAC] section 4.2.2).

For certain, relatively rare events, the client can seek a specific DC, such as the primary DC for the domain. Any such action inherently limits the scalability of the system by forcing traffic through a specific server, and are to be used only under well understood circumstances.

Windows implements two types of domains. This document distinguishes between earlier forms of domains and later AD-based domains by referring to **Windows NT 4.0-style domains** (section 3.1.1) and **AD-style domains** (section 3.1.2). Behavior specific to one form of domain or the other will be called out as such. It is important to note that these groupings are descriptive, and an implementation of these or other protocols around DC functionality can make implementation-dependent decisions about what levels of support or functionality to provide.

The term account database is used to refer to that portion of the **directory** that maintains the accounts for the principal of the domain. The account database can be considered a subset of the complete LDAP-accessible directory that an AD-style domain hosts. The account database comprises all of what is available in Windows NT 4.0-style domains.

Finally, because domains have both traditional client computers (workstations or clients) as well as server computers (servers) as members, the normal terms of client and server as used in other technical documents can become ambiguous. For this document, the term "client" generally applies to any member of a domain, even if that domain member is itself a server computer offering other services to its clients. The term "server" will generally apply to the DC. These terms can be further clarified as **domain client** and **DC server** or DC for clarity in certain sections. It is important to note that this document covers cases where a server is acting as a client to another server. In the cases that one domain client is acting as a supplicant to another domain client, the former is referred to as a **domain client in a workstation role**, and the latter as a domain client in a server role.

Copyright © 2011 Microsoft Corporation.

3.1.1 NT-4.0 Style Domain

A Windows NT 4.0-style domain can be modeled as a simple table of accounts, where each row is a unique account. This type of domain's name is a NetBIOS name resolved by a WINS server. Windows NT 4.0-style domains do not have DNS names, do not have an Active Directory (AD) implementation or many other services that work with AD, and do not support Kerberos as an authentication protocol. When attempting to locate a NT 4.0-style domain the application can only use NetBIOS names.

3.1.2 AD-Style Domain

An AD-style domain is a hierarchical store of many different object types, of which account objects are one type. AD-Style domains have a flat NetBIOS name and an established fully qualified domain name (FQDN). The FQDN is the preferred name for this type of domain. Because of this, when attempting to locate an AD-style domain, applications using NetBIOS names and FQDNs can be run simultaneously. AD-style domains implement Active Directory (AD), LDAP, Kerberos authentication, and advanced configurations and features not supported in NT 4.0-style domains.

3.1.3 Domain Naming

A domain requires a name in order to be useful to clients. An AD-style domain requires a **Domain Name System (DNS)** name; it can also have a flat (non-hierarchical) NetBIOS name. The DNS name of the domain has to be unique--there can be only one domain with a given name. It is an error to configure the system such that domain names conflict. In practice, if domain names conflict, clients can connect to a domain controller (DC) that is not actually serving the domain for which the client was seeking, and thus fail to obtain the services that were desired. If the domain intends to support clients that only understand Windows NT 4.0-style domains, then the domain requires a NetBIOS name as well. Windows NT 4.0-style domains do not support DNS naming, and require a unique (for that network) NetBIOS name assigned for the domain.

By convention, the NetBIOS domain name is the most specific (left-most) element of the DNS domain name, made upper case (although the name is not case sensitive). DNS domain names are limited to 255 characters overall and no more than 63 characters per label within the name, as specified in [RFC1034]. NetBIOS names are limited to 15 characters, as specified in [MS-NBTE]. It is legal to have a domain that has a domain name where the most specific label is longer than 15 characters; that domain, however, will not have a corresponding NetBIOS name that is derived from the DNS name. It may have another name, defined through implementation-specific means. It is also possible to have a less-specific label be longer than 15 characters, with no effect on the NetBIOS name. That is, a domain name such as domain.labelwithmorethanfifteencharacters.com is, in fact, following the conventions.

The NetBIOS name is supposed to be unique as well, but this is not, strictly speaking, required. It is possible to create an arbitrary NetBIOS domain name not related to the DNS name. Also note that DNS is not the same as NetBIOS, and the character sets supported by each naming system are different. Along the same lines as the leftmost element convention, the administrator looks to the intersection of the common characters to both NetBIOS and DNS in order to select the name.

It is also possible to set up development.eastcoast.sample.com and development.westcoast.sample.com on the same network. Because the flat name is by definition ambiguous, services that use that flat name can run into problems. This sort of configuration is never recommended.

3.1.4 Local and Remote Domains

Domains come in two varieties, local and remote.

Copyright © 2011 Microsoft Corporation.

3.1.4.1 Local Domains

Every computer running Microsoft Windows® has a local domain; that is, it has an **account** database for accounts that are specific to that computer. Conceptually, this is an account database like any other with accounts, **groups**, **security identifiers (SID)**, and so on. These are referred to as local accounts, local groups, and so on. Because computers typically do not trust each other for account information, these identities stay local to the computer on which they were created.

3.1.4.2 Remote Domains and Domain Controllers

With a remote domain, certain Microsoft Windows®-based servers can be configured to be domain controllers (DC). A DC is a server that has made its account database available to other machines in a controlled manner. Starting with Microsoft Windows® 2000 operating system, DCs began supporting a database of more than just accounts, becoming a general-purpose directory. This is known as Active Directory.

Because the account database is typically distributed across multiple DCs, there can naturally be a mix of different versions of the individual servers. Active Directory has the notion of a functional level, which serves as a version level for the entire directory. The functional level is managed by the administrator and the system itself.

A domain has built-in groups; these groups are defined by Microsoft and created within the domain during installation. For example, built-in groups include the Domain Users, Domain Computers, and Domain Admins groups. By default, the Domain Users group includes all users who are defined in the domain.

A DC accepts authentication requests on behalf of the machines that have chosen to trust it.

A DC can have peers within the domain. These peers are other servers that also have been configured to host this account database. Any server participating in the domain as a DC may or may not allow changes; the configuration is a choice of the administrator.

When a change is allowed, the servers replicate the change so that all DCs have the same information.

3.1.4.3 Domain Membership

Domain membership is the state of trusting a third party (the domain controller (DC)) for identity and authentication information. Any system can conceivably be part of a domain. Microsoft Windows®-based systems can easily be configured to be part of a domain and trust their DC for many tasks. Also, certain configuration changes are made, such as accepting the domain as the authoritative source of time.

Windows-based systems can have local groups that include members from a domain. This allows the member system to manage its resources in the manner most relevant to it and not be completely dependent on the decisions of the domain administrator. A domain administrator can create a domain local group for each resource that exists within a domain, such as file shares or printers, and then add the appropriate global groups from each domain to this domain local group. The domain administrator then assigns the appropriate permissions for the resources to the domain local group.

Joining a domain ultimately distills down to establishing an account on the domain that represents the system joining the domain, and to setting the password (or key) for the account on both the domain and the actual system. In Windows, this process is encapsulated in a domain join function (**NetJoinDomain**). Several tools, such as **WinBind**, exist for non-Windows operating systems to join a Windows domain.

Copyright © 2011 Microsoft Corporation.

All Windows-based systems have a component that manages their relationship with their DC. This component, called **Netlogon**, maintains the keys that are necessary for ongoing authentication of the member system to the DC. It also creates a general-purpose channel to the **Netlogon** instance on the DC.

This channel is used by various authentication protocol implementations to redirect an authentication request to (or augment their activities with) their instance on the DC.

3.1.4.4 Effect on Accounts

Microsoft Windows® domains have an effect on the way that accounts and groups work. Some of this is by convention and some is by design.

By convention, when a Windows-based system is added to a domain, the domain administrators group is made a member of the local administrators group.<1>

By design, groups have different scopes when domains are involved. Groups can be defined to be globally known and thus usable by other domains or known only within the domain in which they are defined.

3.2 Domain Controllers

In a domain there domain controllers that can perform read and write operations on the directory (writable domain controllers) and domain controllers that can only read the directory to respond to authentication requests from domain clients.

3.2.1 Writable Domain Controller

Writable Domain Controllers (WDC) can perform all read/write operations on the Active Directory (AD), can replicate any changes that occur elsewhere in the domain from other WDCs and contain a complete copy of the directory database including credentials for all accounts. WDCs can only be managed by domain administrators.

3.2.2 Read-Only Domain Controller

Read-Only Domain Controllers (RODC) contains a copy of the directory database and a copy of the SYSVOL folder that contains the Group Policy Objects (GPOs) and logon scripts for client computers. It can respond to authentication requests just like a WDC. The RODC cannot write to or update any databases, can only replicate data from a WDC (for both Active Directory (AD) and SYSVOL), and contains a complete copy of the database except for credentials and other credential-like attributes that are part of the RODC filtered attributes set. RODCs can be administrated by delegated users that do not have any domain privileges beyond standard domain users.

3.3 Accounts

Accounts are always created relative to some issuing authority, which is responsible for allocating and assigning a security identifier (SID).<2>The basis for all interactions with a domain is an identity and proof of that identity. In the case of a domain, the identity is represented by an account in the account database for that domain, and the proof is by demonstrating knowledge of the password, also referred to as a **shared secret**, associated with that account. By knowing both, an entity can establish an authenticated connection to a domain controller (DC). After the DC and a client have authenticated each other, the DC can be used to create authenticated connections to other principals (for more details, see [MS-WSO] section 3.1.2.2).

Copyright © 2011 Microsoft Corporation.

3.3.1 Account Types

There are three main types of accounts: The user account, the service account, and the machine account.

The user account is used to represent a person. The service account is very similar to a user account, but is used to represent a specific identity for an application running on the network. There is very little difference between these two types of accounts, except that the service account will likely have additional names associated with it. For information about names, see section 3.3.2.

Finally, the machine account is used to represent a computer that participates in the domain. In one way, these are simply service accounts with well-known conventions around the name of the account and additional names associated with the account.

Principals in the domain are responsible for knowing the name used on their accounts. For accounts representing people, the person is required to know the name of the account. For a service or machine, some configuration on the client of the domain typically indicates the name that will be used.

In the same way, the person is required to possess the key that is used to authenticate the user account. Typically, this is a password, but it can also be an asymmetric key contained on an external device such as a smart card. Service and machine accounts typically have their keys stored in some form with the configuration; for example, the "keytab" file used in MIT-based Kerberos implementations.

3.3.2 Account Names

Accounts are named in a number of ways, related to the protocol used to access the account. Additionally, the name implies certain attributes about the principal itself, and that has certain implications for how the account is used. All account names serve as aliases to the single account object. The formats are formally defined in [MS-ADTS], but for the purposes of this document, they are as follows.

Distinguished Name: The Distinguished Name (DN) of the account is the LDAP name of the account object. This name is a fully qualified directory path name, starting at the root of the domain's directory. There is no attribute for this name, since this name is the directory's primary way of identifying the object.

SAM Account Name: The Security Account Manager (SAM) Account Name is considered the "flat" name of the account. That is, it is a single string that has no structure. The name is always relative to the domain; two domains can have accounts with the same sAMAccountName value.

User Principal Name: The User Principal Name (UPN) is an alias for the account that is structured as an email address. The UPN can relate to the domain in which the account is located, or it might not. If it does not, it can match another domain or be completely unrelated. The intention is to allow an account to be known internally as account@corp.example.com, but referenced by the user as the more familiar email address account@example.com.

Service Principal Name: The Service Principal Name (SPN) is the name by which a service is identified by a requesting client. The SPN refers to a unique instance of a service, and, when used properly by a client, allows the client to authenticate the server at the same time that the server authenticates the client. The SPN is constructed by the client most often as service/instance (for example, cifs/fileserver.sample.com). The presence of an SPN can be taken as an indication that the account is not associated with a user but rather a service.

Copyright © 2011 Microsoft Corporation.

Alternate Security Identities: The Alternate Security Identities, or altSecurityIdentities, of an account allow other arbitrary protocols to associate names with accounts. The use of altSecurityIdentities is implementation-specific, since it consists of a naming convention between a security protocol server and the account database.

Security Identifier: The security identifier (SID) is a numeric name for the account. The SID is more rigorously defined in [MS-DTYP], but consists of a domain identifier portion that identifies the domain, and a domain-relative portion that identifies the account within the domain.

3.4 Domain Services

Active Directory Domain Services (AD DS) implements one or more domains within a **forest**. A domain provides a number of services to its clients, primarily related to security and management. The **security principals** of the domain are all available from the AD DS **domain controller**, so the domain serves as the primary source of identity for the clients of the domain. The domain, through the relevant security protocols, provides the basis for authentication within the domain, allowing principal within the domain to establish authenticated connections with each other. Once authenticated, the domain provides authorization information in the form of additional identities representing groups, allowing authorization decisions to be made.

AD DS is provided as an optional subsystem of the Microsoft Windows® Server operating system. When installed on one or more instances of the Windows Server, those instances--then known as AD DS domain controllers (DCs)--provide security services to security principals on all network nodes that participate in the AD DS domain. Because of its integration into the operating system security model in Windows implementations, AD DS permits only a single DC to run on a single server. Also, clients can join an AD DS domain, as summarized in Common System Architecture (section 4.3) and [MS-WSO] (section 4.2).

Besides AD DS, Active Directory can operate in a different mode called Active Directory Lightweight Directory Services (AD LDS). Unlike AD DS, AD LDS does not host **domain naming contexts (domain NCs)** and, therefore, does not provide any domain services. It is an application-oriented directory service implemented by one or more DCs within an AD LDS forest. AD LDS is useful to applications requiring read or read-write access to a replicated database containing directory information accessible via the LDAP protocol. In addition, a single system can host multiple AD LDS DCs, whether in the same or in different AD LDS forests. Moreover, unlike AD DS (which installs and operates only on Windows Server), AD LDS can be installed and used on both workstation and server versions of the Windows operating system. Finally, because it does not supply domain services, there is no concept of a client system "joining" a collection of AD LDS servers (an AD LDS forest).

In both AD DS and AD LDS, the directory service provides a data store for objects that typically is distributed across multiple DCs. The DCs interoperate as peers to ensure that a local change to an object replicates correctly across all peer DCs. AD DS, however, uniquely provides integrated authentication and authorization services to security principals operating within its security realm. The most significant content difference between AD LDS and AD DS is that AD LDS does not host domain naming contexts (domain NCs).

[MS-ADTS] contains many more details on AD DS and AD LDS forests, as well as AD DS domains. [MS-WSO] contains further information about how clients are affected by being part of a domain; for a list of affected protocols, see [MS-WSO] (section 4.2).

3.5 Domains and Forests

Domains can be linked together in a number of ways; such a link is termed a *trust*. A single domain that is not linked to any other domain is the sole source of authority for all principals within that

Copyright © 2011 Microsoft Corporation.

domain. Domain trusts, and their arrangements, are again analogous to the cross-realm trust as specified in Kerberos ([RFC4120] section 1.2). How the linking of domains is accomplished is an implementation-specific, server-to-server operation outside the scope of this document.

Domains can be linked in several different ways. Linking domains by establishing a trust relationship between the two domains indicates that the domains trust each other to authenticate the identities of accounts within each domain. The trust need not be complete; the nature of the trust is an administrative decision made by each domain administrator.

Figure 1: Domain trust relationship

Given two domains, A and B, a variety of possible trust relationships can be established. In the preceding diagram, the direction of the arrows indicates that B can trust principals authenticated by A, but A does not trust B. In this case, the direction of the trust is from A to B. Similarly, B could trust A, and accept principals from A's domain, but A might not trust B. And, of course, A and B might have mutual trust, where both sides trust the other side to authenticate principals.

Domains, through their Domain Name System (DNS) names, can be linked to establish trust that mirrors the hierarchy expressed in their names. For example, *child1.example.com* and *child2.example.com* would be configured to trust *example.com*, but neither would trust each other directly. Domains linked this way are termed *trees*. Two or more trees can be linked together to create a forest, where the trust is established at the root of each tree.

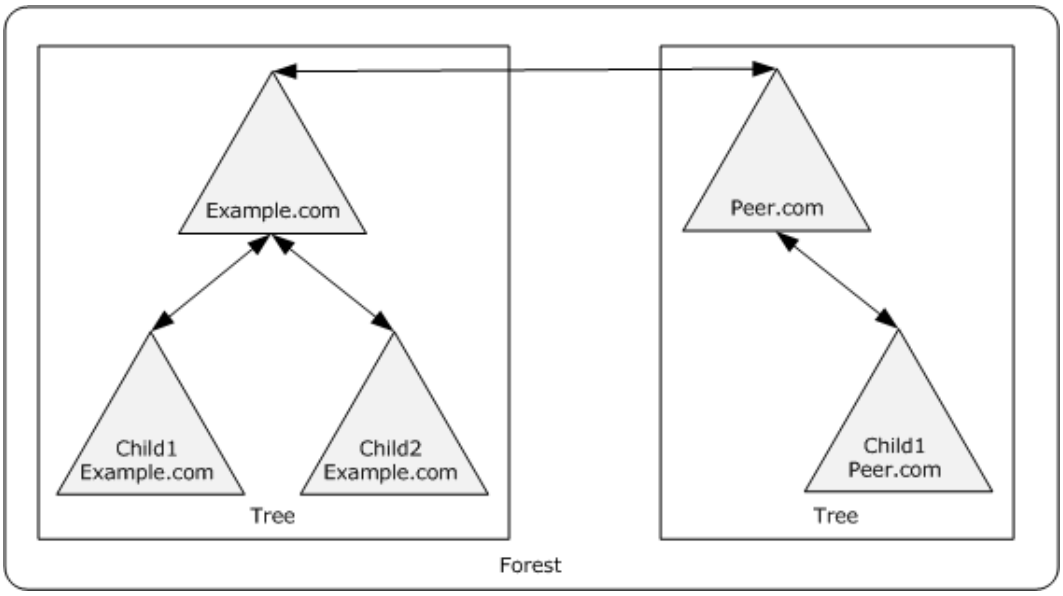

Figure 2: Linked domains creating a forest

Domain trust can be established without a forest. Any two domains can be arbitrarily linked together through a trust. For Windows NT 4.0-style domains, all trusts are established this way.

Copyright © 2011 Microsoft Corporation.

4 Common Task Information

This section contains specifications that are common to many of the other tasks described in this document.

4.1 System Context

The following sections describe system environment elements, assumptions, and preconditions that are common to all the Domain Interactions System Defined Tasks.

4.1.1 System Environment

The Domain Interactions system forms the framework that other systems leverage in their environments. As such, this system requires comparatively little in terms of services available for use, since its purpose is to create a useful environment for other scenarios. Services that this system requires from its environment include the following:

Network Infrastructure. This system requires that a viable network system is available. This includes a networked environment that supports TCP/IP and UDP/IP. Additionally, a name resolution system must be available for use by both domain controller server and **Domain Members**. The name resolution system must support Domain Name System (DNS) form if the domain is to support AD-style domain functionality, and NetBIOS form if the domain is to support Windows NT 4.0-style domain functionality. The system has no requirement for the management model of the DNS namespace, although some form of dynamic DNS will likely be easier to manage.

Coexistence. As discussed in section 3.1.1, any given domain on a network must be uniquely named. There is no architectural limit to the number of domains possible on a network.

Even at this relatively high level, the domain interactions system is a complex aggregation. The relationships among the different systems involved need to be represented first. Once that is established, the interrelations among major components will be far more understandable.

4.1.2 System Assumptions and Preconditions

The heart of the domain interaction system is a set of authentication protocols that form the base upon which many other systems are built. As such, it has very little in the way of assumptions or preconditions.

4.1.2.1 Client

The client assumes basic network connectivity and the availability of basic network infrastructure services such as Domain Name System (DNS).

Prior to being associated with a domain, there are no other preconditions of note. Once a client has been associated with a domain, it has the assumption that the domain controller also has an entry in its directory corresponding to the client. Should this assumption be proven wrong, the system (from the client's perspective) becomes unusable until the association is reestablished.

4.1.2.2 Domain Controller Server

The domain controller publishes its name and capabilities with a DNS infrastructure and/or a **NetBIOS** infrastructure ([MS-ADTS] sections 7.3.2 and 7.3.4). See section 5 (Locating a Domain Controller) for additional details on how the client consumes the DNS records registered by the domain controller. See [MS-ADSO] for additional details on domain controllers.

Copyright © 2011 Microsoft Corporation.

As noted earlier, clients treat all domain controller instances as equivalent. The domain controller has to ensure that it is synchronized with its peer domain controllers, if any are supported in the implementation, through implementation-specific means.

4.2 Common System Relationships

4.2.1 Black Box Relationship Diagrams

The following figure illustrates two possible distributed configurations for the Domain Interactions System, spanning several computers and services in a distributed network.

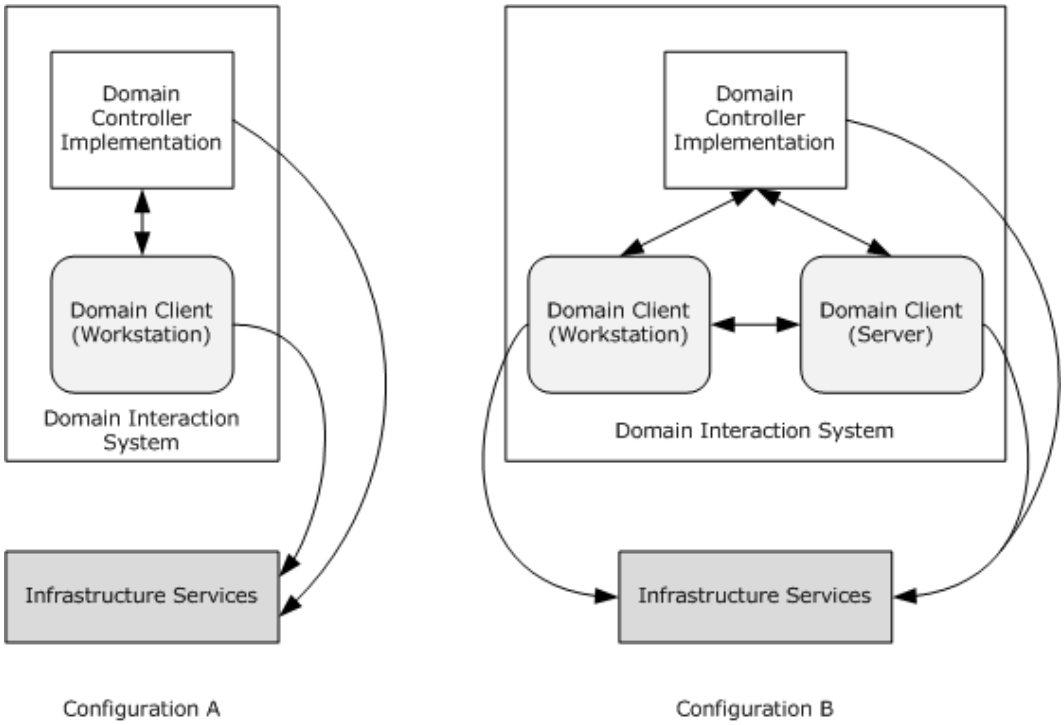

Figure 3: Domain Interactions Distributed System

In this diagram, the Domain Interactions System spans the Domain Clients and the Domain Controller Implementation, but also interacts with the independent infrastructure services. The infrastructure services include services such as name resolution (DNS, WINS) and network maintenance services (routers). The Domain Controller Implementation uses such services to make itself available to its clients.

The clients are shown in two different configurations labeled A and B. In the configuration A, the Domain Client (Workstation) is simply the consumer of the services offered by the domain controller implementation. An example of a domain service is interactive logon to enable a user to logon interactively into the Domain Client (Workstation).

Configuration B has two clients of the domain, one labeled Domain Client (Workstation) and the other tagged Domain Client (Server). In this configuration, the Domain Client (Workstation) is consuming services offered by the Domain Client (Server), as indicated by the double-arrowed line. Regardless of the specifics of those services, both parties are relying on the Domain Controller Implementation for services that enable those of the server itself. These services typically would be supporting authentication or authorization.

Copyright © 2011 Microsoft Corporation.

4.2.2 Common System Dependencies

Due to the nature of the Domain Interaction System being distributed among many computers for different, but related, purposes, enumerating the dependencies of the system is also complex. The coarse diagram in the following figure serves as a very high-level description of how the dependencies among the components can be visualized.

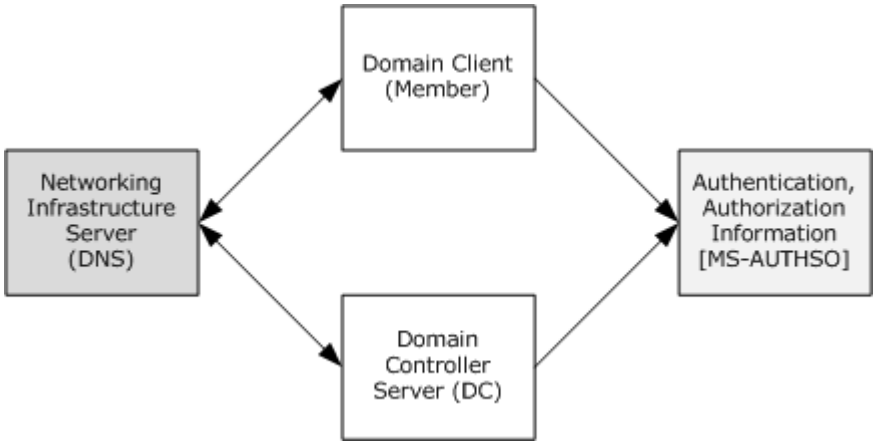

Figure 4: Dependencies between Domain Interaction Distributed System and other components

In this diagram, the dependencies of the system are very symmetrical between the domain client and the domain controller server. Both the domain client and domain controller server rely upon infrastructure servers such as DNS, and leverage those servers for locating each other (rendezvous). During this rendezvous process, the domain controller server publishes its name and the domain client locates the domain controller server through DNS. The details of this rendezvous process are described in section 5.

In addition to service location, the rendezvous process between a domain client and a domain server relies upon authentication and authorization information. The domain controller server, for example, will leverage the authorization information that it contains for controlling access to its resources. A more complete description of this authentication and authorization information can be found in [MS-AUTHSO] and related documents.

4.2.3 Common System Influences

Because the Domain Interactions System specifies how systems and computers function within a domain environment, any system or protocol that can operate within a domain, or has a mode of operation within a domain, is influenced by this system. However, there are a few Protocol Family and Defined Task systems where the influence is best called out more explicitly:

Active Directory System ([MS-ADSO]): has a more in-depth description of how the directory is structured, and how LDAP operations can be made.

Authentication System ([MS-AUTHSO]): specifies how other protocols take advantage of the authentication protocols such as NTLM or Kerberos to secure their communications, and the authentication services that support the client to server communication. [MS-AUTHSO] depends on DIS for specifying how those protocols are used in a domain context, to authenticate clients to servers when both are members of a domain.

Copyright © 2011 Microsoft Corporation.

A series of families leverage the domain controller as the source of identity and authorization information for the domain. These include:

Certificate Services System ([MS-CASO]): specifies how the certificate authority leverages the domain infrastructure to manage certificate distribution and enrollment, and makes authorization decisions based on information associated with the accounts in the domain.

Rights Management System ([MS-RMSO]): specifies how content can be protected against offline access, based on authorization information from the directory.

Transaction Processing System ([MS-TPSO]): specifies how transaction processing can be synchronized across multiple machines, based in part on authentication and authorization services received from the domain.

File Services ([MS-FSSO]): specifies how file servers present a unified view of files and other resources, and rely upon [MS-AUTHSO] and DIS for authentication when the file server is part of a domain.

Print Services ([MS-PSSO]): specifies how print servers can render content, and rely upon DIS for receiving authorization information about print operations when the print server is part of a domain.

Message Queuing Services ([MS-MQSO]): specifies how operations can be queued, and relies upon DIS for authentication and authorization when the queue participants are part of a domain.

Network Policy and Access Services ([MS-NAPSO]): specifies how machines can be examined for access to a network. The machines have to be members of a domain in order to authenticate to the NAP servers.

Finally, the Domain Interactions system forms the basis for managing resources within the domain. Thus it influence the following families as well:

Windows Management Services System Overview ([MS-WMSO]): specifies the set of protocols used to manage servers remotely; when those servers are part of a domain, this relies upon DIS for authentication and authorization services.

Group Policy System Overview ([MS-GPSO]): specifies how domain clients can retrieve group policy information from the domain controller, which is based on the group memberships of the domain accounts, as well as the domain account's location in the LDAP directory structure.

Windows Server Update Services ([MS-WSUSO]): specifies how different machines in a domain can have different update policies for patch management, which relies upon DIS to specify the domain authorization information.

4.3 Common System Architecture

The following sections describe the abstract client-side and domain-controller side data model elements that apply to all Domain Interactions System Defined Tasks.

4.3.1 Common Abstract Data Model

This section describes common state established, used, and maintained by processing rules of Domain Interactions Services Defined Tasks. State can be volatile or persisted. State can pertain to one, some, or all instances of the task. The overall organization of the data elements, with their names, is the Abstract Data Model. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules might depend upon associations established by the structure of its Abstract Data Model, such association can be achieved in other ways.

Copyright © 2011 Microsoft Corporation.

Implementations can depart from this model so long as their external behavior remains consistent with that described in this document.

The Abstract Data Model (ADM) is best described by separating the client of the domain from the domain controller. The client's model is fairly simple; the domain controller's model is somewhat more complex.

4.3.1.1 Client Data Model

The client's data model is fairly simple. The client must maintain the following in a persistent store. The following elements must be shared in a read-only mode with other protocols on the client unless otherwise specified, and are shared in a read-write mode with client administrators. For information about other protocols that refer to these ADM elements, see [MS-ADMS] section 6.

DomainName (Public): The client MUST know the name of the domain to which it belongs. The **DomainName** is further defined as **DomainName.FQDN (Public)**, indicating the fully qualified Domain Name System (DNS) domain name, and **DomainName.NetBIOS (Public)**, indicating the NetBIOS name of the domain. Whether **DomainName.FQDN** is present depends on configuration. As noted earlier, Windows NT 4.0-style domains do not have DNS names, so the **DomainName.FQDN** need not be established in that case. For AD-style domains, this is the flat NetBIOS name of the domain.

When the client is not joined to a domain, **DomainName.FQDN** is set to NULL and **DomainName.NetBIOS** is set to the NetBIOS name of the workgroup the client is associated with.

DomainSid (Public): The client MUST preserve the security identifier (SID) of the domain to which it belongs. This SID is used later as part of the authorization process. If the client has never been joined to a domain, or was previously joined then unjoined, this value is empty.

DomainGuid (Public): The client MUST preserve the **GUID** identifier of the domain to which it belongs. If the client has never been joined to a domain, or was previously joined then unjoined, this value is empty.

ForestNameFQDN (Public): The client MUST preserve the canonical fully qualified DNS name of the forest containing DomainName. If the client has never been joined to a domain, or was previously joined then unjoined, this value is empty.

SiteName (Public): The client can retain the **site** that it has determined either through administrative configuration or dynamic discovery. Preserving the site name allows the client to use the site in the process of finding a "near" domain controller (DC) during the location process. However, for clients that are mobile and may shift sites frequently (for example, a business traveler using a laptop), preserving the site may not help, or may require additional information such as network awareness that are outside the scope of this document. Client implementations SHOULD incorporate site awareness and preserve the name of the site.<3>

ClientName (Public): The client MUST know the name of itself, as the domain knows it. This corresponds to the **SAMAccountName** attribute of the object in the directory. The **ClientName** may be populated from configuration (for example, a service or machine name), or from human interaction.

ComputerName (Public): The client of the domain MUST know the name of the computer upon which it is executing, in a form that can be resolved by the underlying network infrastructure. The **ComputerName** element is conceptually the same as the "hostname" element used in other standards; specifically, the name by which the computer can be referenced. The **ComputerName** element has two sub elements, **ComputerName.FQDN (Public)** and **ComputerName.NetBIOS (Public)**. The FQDN sub element refers to the canonical fully qualified DNS name of the computer,

Copyright © 2011 Microsoft Corporation.

and MUST NOT be an alias (such as a CNAME in DNS) to another name. The NetBIOS element is the NetBIOS name of the computer. Either, or both, of these elements may be present, depending on configuration.

The **ComputerName.NetBIOS** element is typically used for holding the NetBIOS name of the computer, and it MUST be a Unicode UTF-8 string [RFC3629]. The NetBIOS name of the computer is typically the same as the unqualified name of the computer (for an example, see use of "simple-name" in [RFC819]), as long as the name fits within the NetBIOS naming constraints. If the simple name does not meet the requirements of NetBIOS host names, then the transformation from simple name to NetBIOS name is an implementation-specific detail. A host participating in this system is not required to implement NetBIOS to interact correctly with other services in the system, although the system does require the use of a flat, unqualified name for the computer or host. For clarity, that will still be referred to as the NetBIOS name, even if the implementation does not use NetBIOS.

Password (Public): The client must know the password credentials associated with the account object for **ClientName** in the directory.

These elements will provide the basis for how the client invokes the protocols used when communicating with the DC. They must be persisted in some implementation-dependent fashion when the client of the domain is not interactive. That is, if the domain client is acting on behalf of a user, it is possible to prompt the user for this information. If the domain client is acting on behalf of a service or set of services (for example, a server), then the implementation must store these values in a way that allows the domain client to retrieve these values.

TrustedDomains: When the domain client is interacting with a DC, it builds a list of domains trusted by the client's DC. This list may be empty if there are no other domains that are trusted. This can be modeled as a simple array, where each element of the array has a domain name, as well as a flag indicating whether the domain is an AD-style domain or a Windows NT 4.0-style domain. The domain name is further broken down into a flat NetBIOS name, **DomainName.NetBIOS**, and a fully qualified DNS name, **DomainName.FQDN**. Either, but not both, of these elements may be empty, based on the type of domain represented by this trust.

4.3.1.2 Interaction with the [MS-LSAD] Data Model

If the client is running the [MS-LSAD] protocol, the following ADM elements (presented in section 4.3.1.1) MUST be considered to be owned by the [MS-LSAD] protocol:

- **DomainName.NetBIOS**
- **DomainName.FQDN**
- **DomainSid**
- **DomainGuid**
- **ForestNameFQDN**

Furthermore, when the client is running the [MS-LSAD] protocol, access to these [MS-LSAD] ADM elements MUST be implemented as described in [MS-LSAD] section 3.1.1.10.

4.3.2 Domain Join State

[MS-ADTS] section 7.4 describes the domain-joined state for a client. [MS-DISO] specifies in more detail the processing logic through which this state is achieved. The client may be either joined to a domain, or not joined to a domain; the client can only be a member of one domain at a time. Transitions between these states can be accomplished as shown in the following diagram:

Copyright © 2011 Microsoft Corporation.

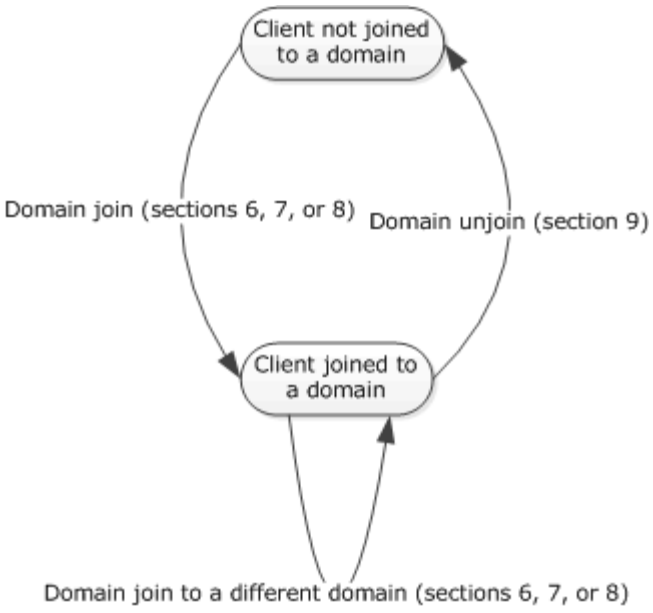

Figure 5: Domain join state

4.4 Overview of the Interactions in the System

There are a number of relationships among the protocols that make up the domain interactions system. In addition to the general theme of this document in dividing the descriptions between the domain client and the domain controller, it is also useful in some areas to distinguish between the roles of the domain client systems. Consider the following figure as a broad map to lay the groundwork for the interactions and relationships between domain clients and domain servers.

Copyright © 2011 Microsoft Corporation.

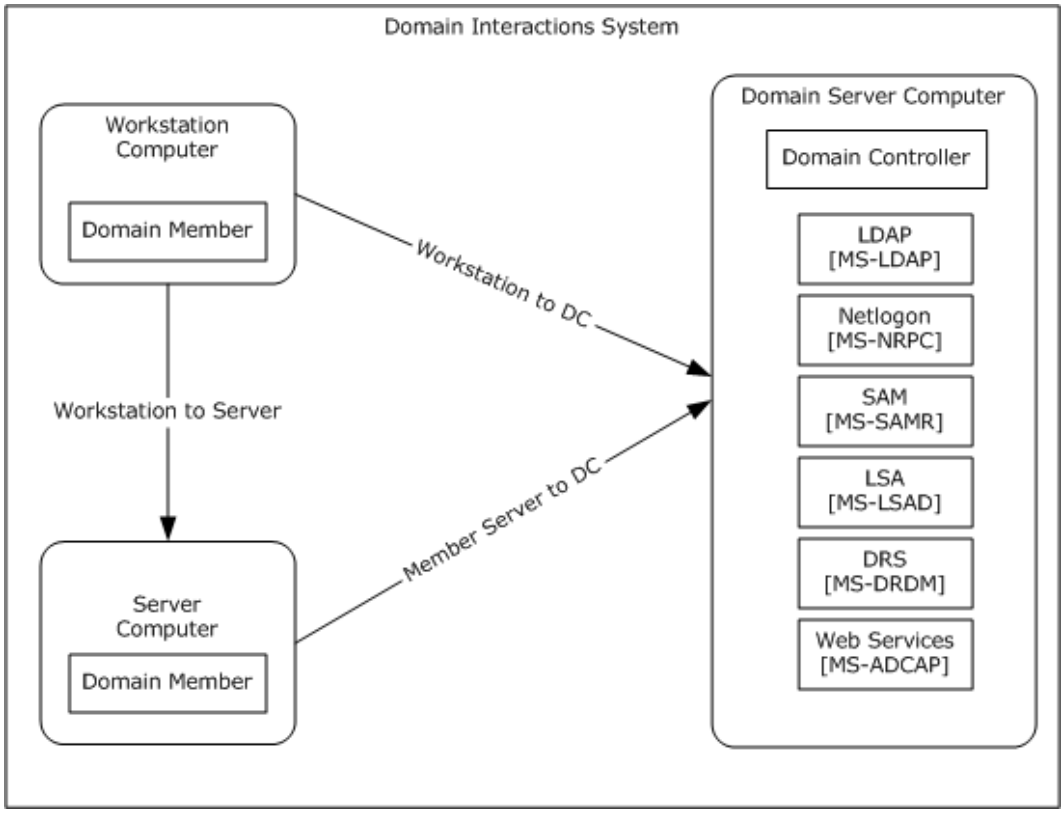

Figure 6: Domain interaction

In this figure, the boxes on the left each represent domain clients, but in different roles. The top box represents a client workstation that is associated with the domain. The bottom box represents a server of some other set of services, but still a client of the domain services (for example, a file server). Each of these clients has a set of relationships with the domain controller, represented by the solid arrows. The details of the relationships between domain members and **domain controllers** are discussed in section 4.5. The communication between the client workstation and the server is not covered in this document, but rather is specific to the services that the server is offering. Communication related to domain interactions between all three machines in the diagram will often require authentication, which is covered in more detail in [MS-AUTHSO].

4.5 Common Relationships in Domain Client Workstation and Server Roles

Even with the taxonomy outlined in the preceding diagram, there is a substantial amount of commonality in being a client of the domain controller. This section describes several simple views of the relationships that are common to both the workstation and server roles.

Copyright © 2011 Microsoft Corporation.

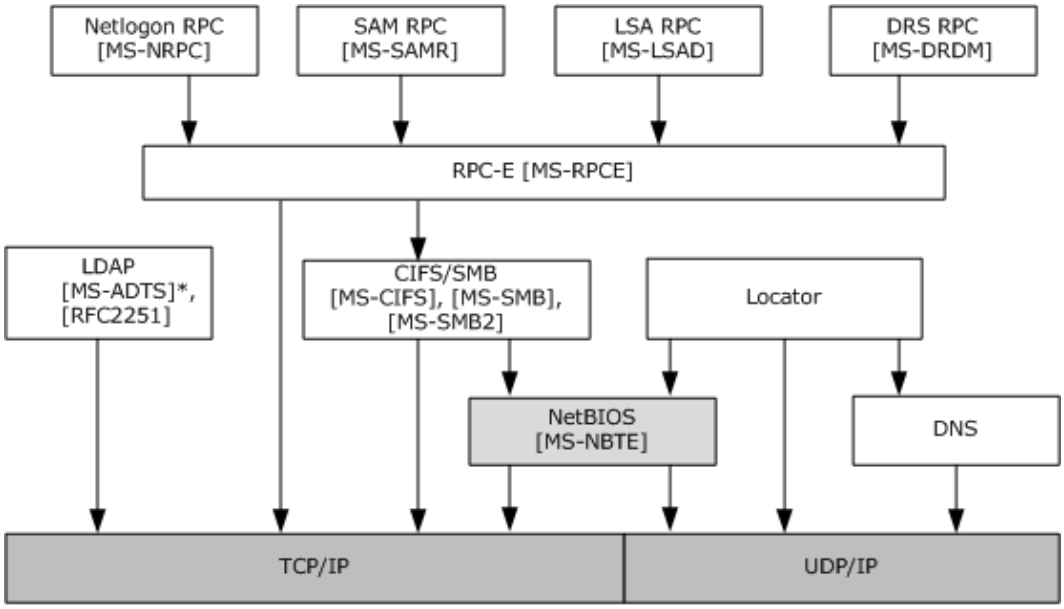

Figure 7: Protocol transport dependencies

The preceding figure shows a view into the transport dependencies of some of the protocols in this family. The darker gray boxes are the transport protocols that form the basis for all communication. For example, this shows that the Locator protocol that is used for locating domain controllers depends on the ability to send UDP packets directly, as well as using NetBIOS for name resolution and broadcast. This diagram is not exhaustive, as it shows only some of the common elements.

Section 6, "Joining a Domain using Predefined Account", depends on SMB (may be [MS-CIFS], [MS-SMB], or [MS-SMB2]), [MS-NRPC], [MS-LSAD], [MS-DRDM], LDAP (as further specified in the LDAP-related parts of [MS-ADTS]), and [MS-SAMR].

Section 7, "Joining a Computer by Creating an Account via SAMR", depends on SMB (may be [MS-CIFS], [MS-SMB], or [MS-SMB2]), [MS-NRPC], [MS-LSAD], [MS-DRDM], and [MS-SAMR].

Section 8, "Joining a Computer by Creating an Account via LDAP", depends on [MS-NRPC], [MS-DRDM], LDAP (as further specified in the LDAP-related parts of [MS-ADTS]), and [MS-SAMR].

Section 9, "Removing a Domain Member", depends on SMB (may be either [MS-CIFS], [MS-SMB], or [MS-SMB2]), [MS-NRPC], and [MS-SAMR].

See individual sections for specifics on the above dependencies.

4.5.1 Workstation to Domain Controller

In addition to the common protocol stack and patterns discussed above, the workstation, or initiator, role adds certain other views. As above, the following figure shows the additional transport dependencies in the workstation role.

Copyright © 2011 Microsoft Corporation.

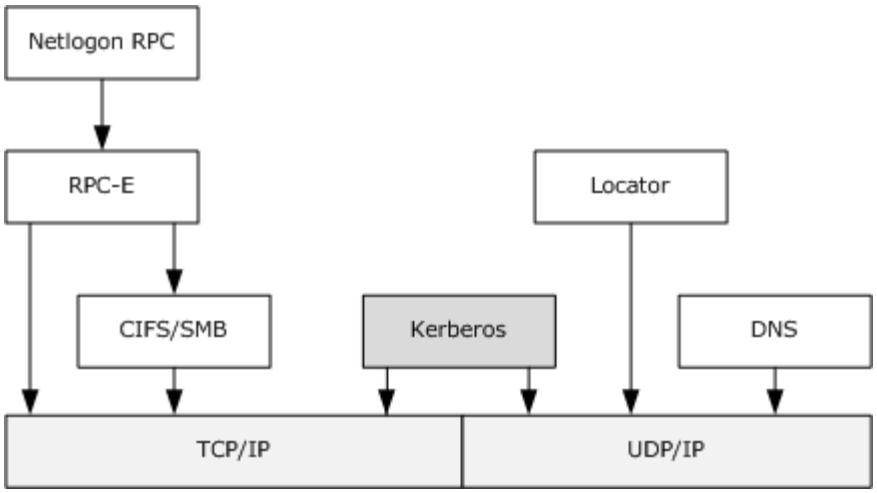

Figure 8: Additional transport dependencies in the workstation role

Here, the Kerberos protocol is shown using both TCP/IP and UDP/IP. The NetBIOS protocol has been removed for clarity.

For a domain member that is acting in the workstation or initiator role, the majority of the functionality is captured in the common case above. The primary difference is that the domain client will use Kerberos for authentication. The same pattern as noted above, where the locator finds a domain controller with a KDC and updates the ADM *DomainController* element, is used in this case as well.

4.5.2 Server to Domain Controller

The server role adds a different aspect because the server (as a client of the domain controller) is bound to forward requests relating to authentication and authorization to the domain controller for processing. These requests ride on top of the [MS-NRPC]: Netlogon Remote Protocol Specification, as shown in the following figure.

Copyright © 2011 Microsoft Corporation.

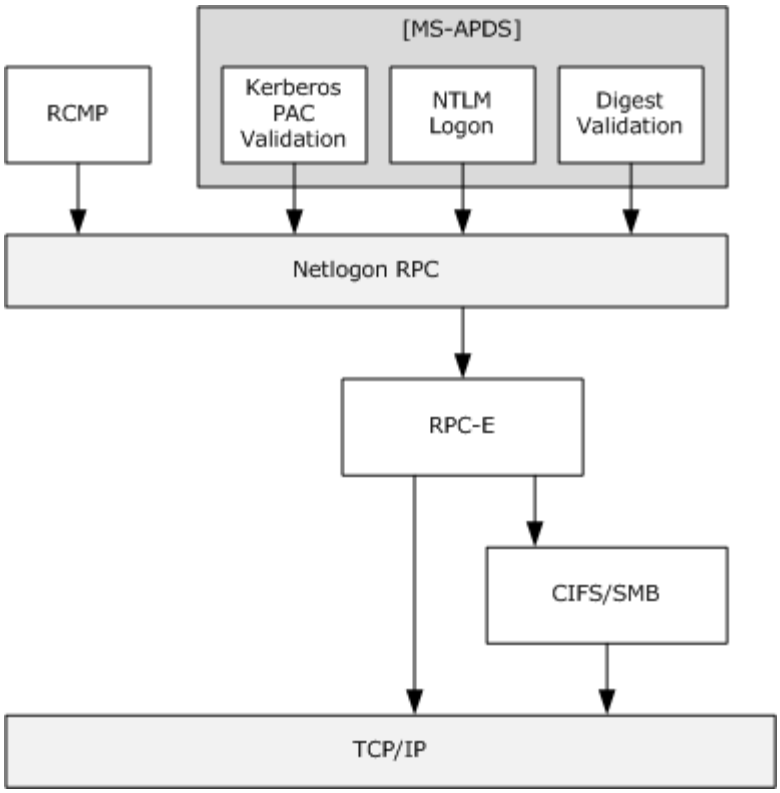

Figure 9: Server requesting authentication and authorization with Netlogon RPC

In this instance, the server can authenticate its client and obtain authorization information from the domain controller by forwarding authentication information (received from the client) to a domain controller using one of the four protocols depicted in the preceding figure. Which protocol the server will use for communication with the domain controller is determined by the authentication protocol between the client and the server as follows.

Protocol between client and server	Protocol between server and domain controller
Remote Certificate Mapping Protocol	Remote Certificate Mapping Protocol
Kerberos Authentication Protocol	Kerberos PAC Validation
NTLM Authentication Protocol	NTLM Logon
Digest Authentication Protocol	Digest Validation

For further information about the use of these four protocols, refer to the following documents: Remote Certificate Mapping Protocol [MS-RCMP], Kerberos PAC Validation [MS-APDS], NTLM Logon [MS-APDS], Digest Validation [MS-APDS]. See section 6 Appendix A Product Behavior of each of the referred documents for a detailed accounting of the protocols supported by different versions of Windows.

Copyright © 2011 Microsoft Corporation.

4.5.3 Domain Controller and Domain Client Functional Relationships

This section covers the relationships among protocols in more detail. As has been noted earlier, the complexity of the system is such that a coarser overview diagram helps set the stage for more detailed diagrams later. While it is often useful to show the entire system, this portion is broken into the server (domain controller) and client views.

4.5.3.1 Domain Controllers

The domain controller exposes the directory through sets of services. The aggregation of these services comprises the "surface" of the domain controller. Each of these groupings is detailed more later in this section. It is important to note that the majority of the state is preserved in the directory database, which is implementation-dependent.

The elements of the ADM for the domain controller side of the domain system are all logically persisted in the database that stores the domain information.

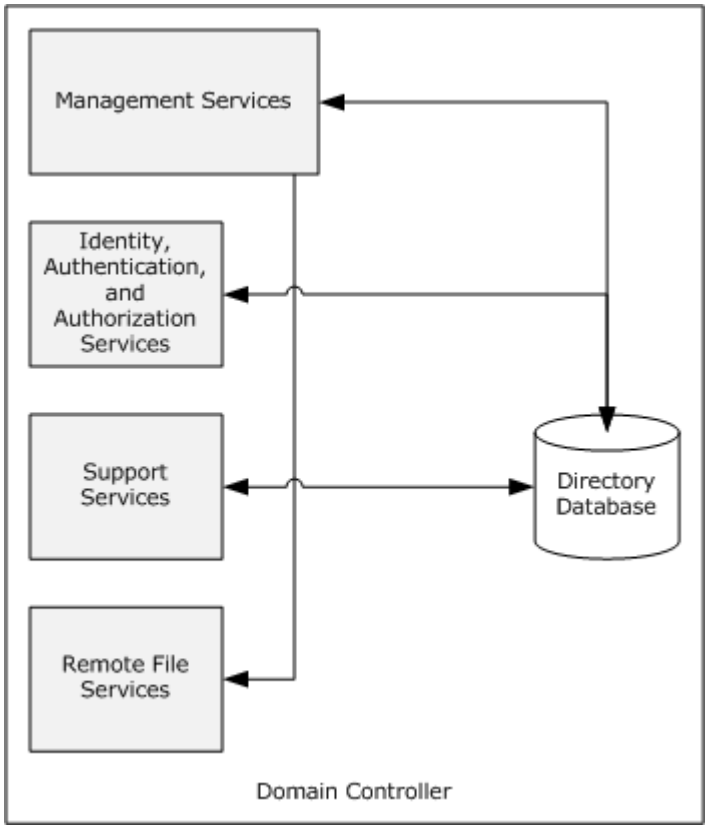

Figure 10: Service interacting with directory database for domain information

4.5.3.1.1 Management Services

Management services are services that are invoked to manipulate the state of the directory, such as adding, deleting, or modifying objects within the directory. For information about using the management services, see [MS-ADSO]. Conceptually, the management interfaces align as shown in the following diagram.

Copyright © 2011 Microsoft Corporation.

Note These interfaces require and use authentication services; however, the services were omitted from this diagram in order to focus on the larger scale view of the system.

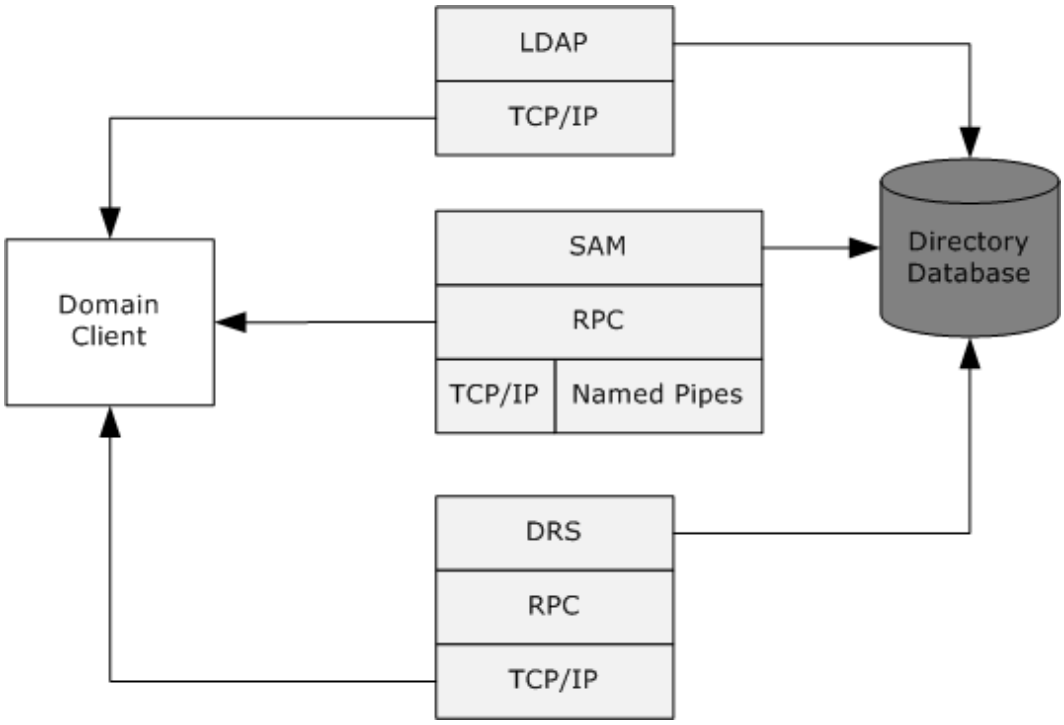

Figure 11: Management services between the domain client and the directory database

- LDAP ([MS-ADTS]). LDAP is the primary method of manipulating objects in an AD-style directory.
- Security Account Manager RPC ([MS-SAMR]). The MS-SAMR RPC protocol is a legacy RPC interface from Windows NT 4.0-style domain implementations. It can only manipulate a subset of the objects that are present in an AD-style domain, but MUST be present for full support of clients. This RPC interface is used primarily over TCP/IP, as well as SMB Named Pipes.

Domain Replication Services RPC ([MS-DRDM]). A subset of this RPC interface is used by clients to translate names of objects in the directory to different formats, and to manipulate the names of certain objects.

4.5.3.1.2 Identity, Authentication, and Authorization

The Identity, Authentication, and Authorization, or IA&A, service group are the services directly related to the identification, authentication, and authorization of principals from the directory. Identity stems from the existence of principals in the domain. Authentication proves the identity of the principal. Through group membership information maintained by the directory, authorization information can be derived and returned to the client of the domain.

Copyright © 2011 Microsoft Corporation.

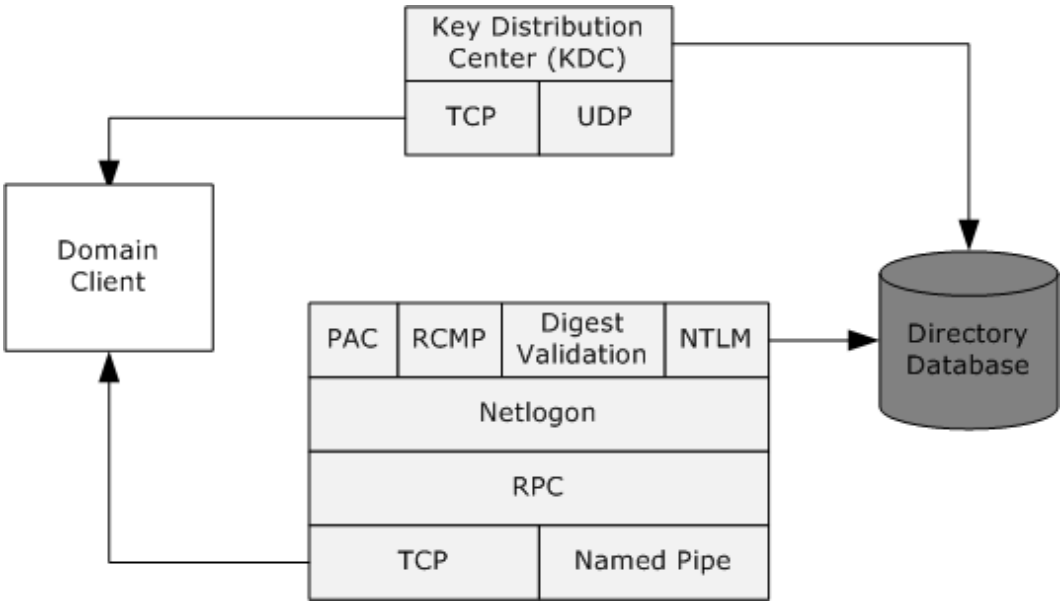

Figure 12: IA&A services between the domain client and the directory database

This figure shows the Kerberos protocol stack and the Netlogon stack, and their relationship to the clients and the domain directory. The Netlogon RPC interface in turn provides a channel for other authentication protocols to execute their authentication, authorization, or other activities at the domain controller. Here, those additional protocols are shown on top of the Netlogon stack.

- Kerberos ([MS-KILE], [RFC4120], etc.): Kerberos is the primary authentication protocol for AD-style domains. The Kerberos Key Distribution Center (KDC) authenticates principals based on the Kerberos protocol, using the directory as the back-end store for account information.
- Netlogon RPC ([MS-NRPC]): Netlogon RPC serves as the general channel by which a client of the domain connects to a domain controller for the purposes of authentication, authorization, or related security purposes.
- NTLM/Netlogon ([MS-NRPC], [MS-NLMP]): NTLM is the secondary authentication protocol for AD-style domains, and the only authentication protocol for Windows NT 4.0-style domains. NTLM authentication is forwarded to a domain controller over the Netlogon RPC interface, allowing servers to authenticate clients without having a complete account store local to the server. As in Kerberos, the directory is used as the account store for authentication.
- Remote Certificate Mapping Protocol/Netlogon ([MS-RCMP]): Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are authentication protocols based on the use of X.509 public key certificates. The association between the certificate and an account is done via the Certificate Mapping protocol, which travels over the Netlogon RPC protocol. The directory is used as the account store.
- Digest Validation ([RFC2617], [RFC2831], [MS-DPSP], [MS-APDS]): Digest authentication is an authentication protocol used primarily in HTTP authentication, but occasionally in LDAP-accessible directories as well. Digest authentication is similar to the NTLM [MS-NLMP] protocol, and validation can take place at the domain controller instead of on the domain client that is handling the server end of the Digest authentication. In much the same way as NTLM over Netlogon is handled, the Digest authentication can be forwarded to the domain controller over the Netlogon channel.

Copyright © 2011 Microsoft Corporation.

- Kerberos PAC Validation ([MS-PAC], [MS-APDS]): The Privilege Attribute Certificate or PAC that is carried in the Kerberos tickets is documented in [MS-PAC]. The PAC is digitally signed by the issuing KDC using a method covered in [MS-PAC] section 2.8.1. A domain member can validate the signature by forwarding the PAC to the domain controller, again over the Netlogon RPC channel. From the user's PAC, Microsoft Windows® identifies the user's principal ID and group memberships and populates the user's security token with that information. The information is subsequently used in access control decisions.

4.5.3.1.3 Support Services

Domain controllers, due to their availability and centrality, may host additional services that are not critical to the functioning of the domain. Windows Clients will look by default to domain controllers to host these services, and may require additional configuration for implementations that choose not to produce them. These include:

- Simple Network Time Protocols (SNTP): Domain Controllers can offer time services to clients of the domain through SNTP [RFC1769]. This can be convenient since Kerberos, among other protocols, requires clocks to be reasonably closely synchronized. The extensions to SNTP for domains, [MS-SNTP], allow the time synchronization messages to be signed using the password from the account in the directory.
- Domain Location: While this is not, strictly speaking, a protocol, the locator functionality as specified in [MS-ADTS] section 7.3 shows what information a domain controller must publish as well as what steps a domain client should take to locate a domain controller.

4.5.3.1.4 Remote File Services

Domain controllers must also act as file servers for several reasons. First, several of the protocols above run over named pipes, which is a construct of SMB/CIFS. Second, as part of the logon process, clients may be returned a logon script to be executed; that script is by default relative to the domain controller. Third, group policy is implemented, in large part, by accessing files from the domain controller. Protocols include:

- SMB/CIFS (as specified in [SMB2], [MS-SMB], and [MS-CIFS]): SMB/CIFS is the primary protocol used for file sharing from the domain controller, and for implementing named pipes for RPC.

  The **client** and the domain controller will negotiate the use of [MS-CIFS], [MS-SMB], or [MS-SMB2] (see [MS-CIFS] section 3.4.4.7 for details). The Tasks in this document do not require that any specific version of the SMB protocol be negotiated, and the term "SMB/CIFS" is used as a generic reference to any of these three protocols.
- DFS ([MS-DFSC]): Domain controllers can act as DFS roots in certain scenarios.

It should be clear that the Management and IA&A services are providing multiple views into a single database of objects, as noted in the ADM above. An implementation of the domain controller services MUST make all views from different protocols onto the database consistent.

4.5.3.2 Domain Client

The domain client comprises a matching set of interfaces, but a substantively different interconnection among the protocols. On the client, information that results from one protocol is often used to seed another protocol's initial state. These interrelations are described later in section 4.7.1.

Copyright © 2011 Microsoft Corporation.

4.6 Common Architectural Details

4.7 Architectural Details

As per earlier sections of this document, it is generally useful to view this system in two ways: as the client of the domain services, and as the domain controller itself.

4.7.1 Domain Client Architecture

The domain client is largely focused on making requests of the domain controller. The general state model of the client is as follows.

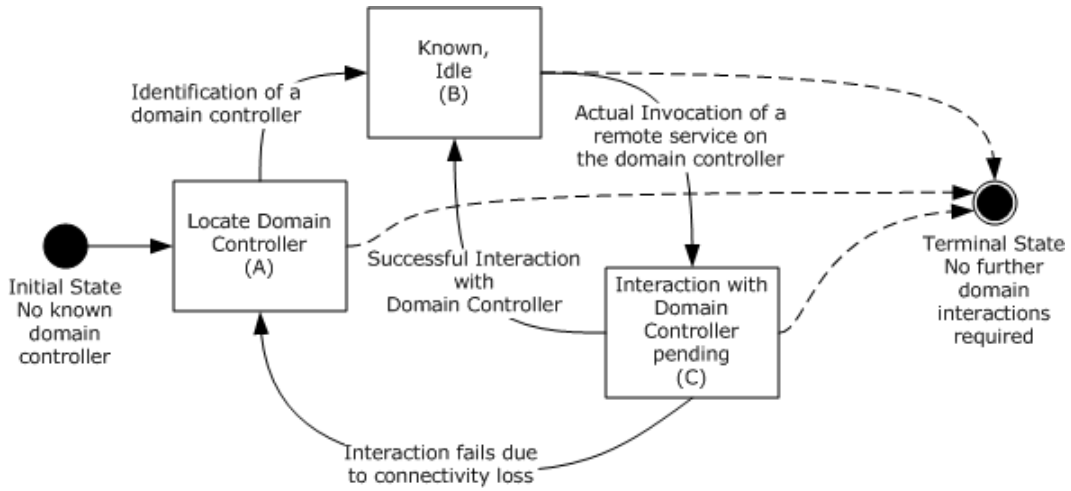

Figure 13: Client state model

Note that this does not attempt to represent the states of individual protocols in the family. This is a very high-level view of the state of the system. In this diagram, the solid lines represent state transition (detailed as follows), and the dashed lines represent transitions to the end state, where the client is shut down. Shutdown may take place for any number of reasons, from user desire to implementation-dependent failure modes. Transitions to the shutdown state are included for completeness, but are not detailed.

There are three main states for the domain client. The first state is "Locate Domain Controller (A)" where the client is using the Locating a Domain Controller task (section 5) in order to locate the domain controller. Upon successful identification of a domain controller, the system transitions to the "Known, Idle (B)" state. Until the domain controller is located, the system cannot move out of the "Locate Domain Controller" state.

Once the system has moved to the "Known, Idle" state, this indicates that the domain controller is known, and that requests of the domain controller can be made. When the system makes a request to the domain controller, the system moves to the "Interaction with Domain Controller pending (C)" state. In this state, the system has made a request, and is now waiting for the domain controller to reply. This state has two main exit paths. The first is that the request has completed successfully. Note that this does not imply that the request was granted, merely that the request was answered. Once the answer has been obtained, the system moves back to the "Known, Idle" state. Since the request was answered, the domain controller is still valid.

Copyright © 2011 Microsoft Corporation.

Should no answer come back from the domain controller, then the request is canceled, and the state of the system reset to "Locate Domain Controller". This transition indicates that no domain controller is currently available, and that the system must wait until a domain controller can be found again.

As noted above, the client architecture is somewhat complementary to that of the domain controller. The groupings of protocols are aligned, but the shared state between the groups and protocols is different. This shared state is captured in the following figure.

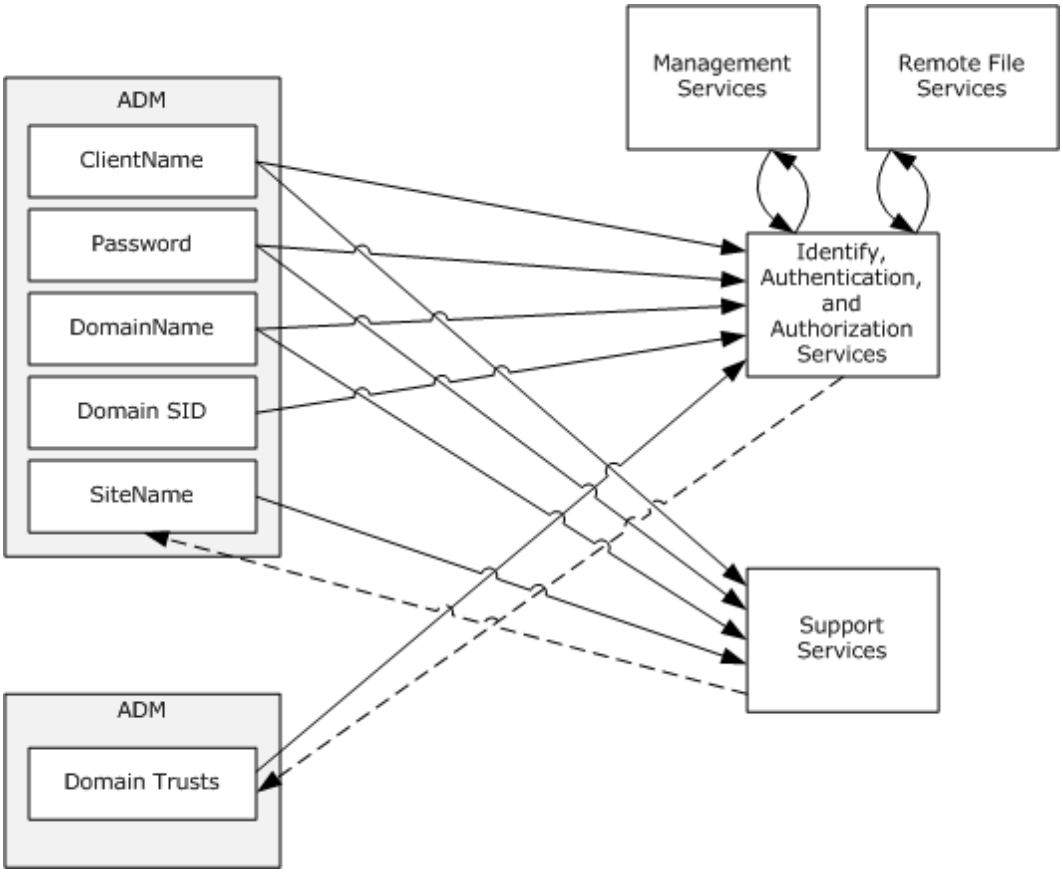

Figure 14: Shared state between groups and protocols

In this diagram, the persistent portions of the ADM are in the upper ADM box, and the transient portions are in the lower box. Updates are expressed as the dashed straight line, and simple consumption of the element is a solid straight line. Encapsulation of other services is represented with an arc.

4.7.1.1 Locator

The Locating a Domain Controller task (section 5) is responsible for locating a suitable domain controller.

The locator performs a name lookup with the DNS infrastructure based on the DNS domain name to identify a list of candidate addresses for domain controllers. Subsequently, the LDAP ping mechanism ([MS-ADTS] section 7.3.3) is used to determine if a domain controller from that list is an appropriate domain controller to use (see section 5.4.5.3 for details). Alternatively, if only the NetBIOS domain name is known, a MAILSLOT Ping ([MS-ADTS] section 7.3.4) mechanism is used to

Copyright © 2011 Microsoft Corporation.

locate an appropriate domain controller using the Remote Mailslot Protocol (see section 5.4.5.5 for details).

4.7.1.2 Time Service

The time service is an excellent example of how a simple protocol such as SNTP is integrated into the domain functionality. The domain client has an SNTP client locally that is tasked with managing the system clock. It is important that the clock of the domain client be reasonably closely synchronized with the domain controller's clock. The Kerberos protocol, for example, requires that the clocks be synchronized to within five minutes of each other.

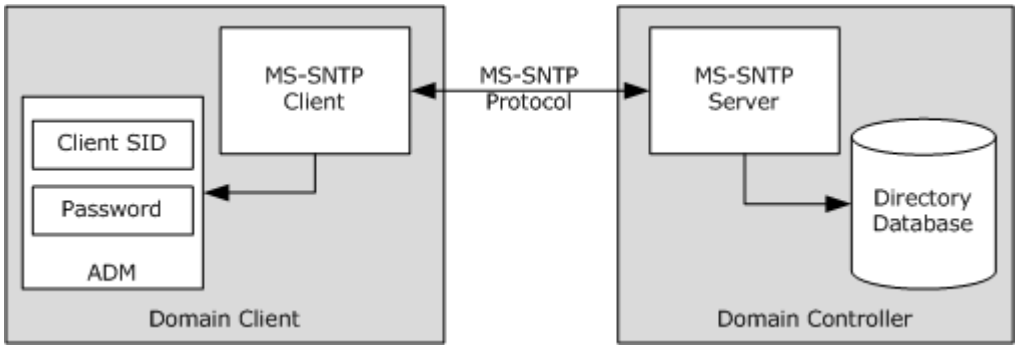

Figure 15: Example protocol utilizing domain client ADM

This SNTP client on the domain client is configured, by default, to use a domain controller as its source of time. The SNTP client invokes the locator to identify a candidate domain controller. The time service also uses the RID of the client computer's account in the domain as the Key Identifier in the client NTP request (see [MS-SNTP] section 2.2.1). The SNTP server can use this RID to look up the account for the domain client in the directory. From the directory, the password associated with this account can be used to create a cryptographic check sum of the time stamp for return to the client ([MS-SNTP] section 2.2.2). The SNTP client on the domain client can then verify this checksum, based on the **Password** element of the ADM.

This example shows how a protocol leverages the same ADM elements as part of the operation of the domain client.

4.7.1.3 Authentication

The Identity, Authentication and Authorization set of protocols consumes ClientName, Password, and DomainName, and makes use of DomainController.Address. However, this set of protocols is typically used via encapsulation, where, by way of example, SMB/CIFS within the Remote File Service group would make an authenticated connection to a server, or the LDAP binding to the domain controller would be authenticated. These cases are covered in greater detail in [MS-AUTHSO].

The IA&A group is used independently of other encapsulating protocols when servicing an authentication operation such as interactive logon.

4.7.2 Domain Controller Architecture

The majority of the interactions with the domain controller are transitory, in that the client selects a task, binds to the protocol group on the domain controller, performs the task, then unbinds. In those cases, the domain controller server functions much as the server in any other task, responding to the varied requests of the clients.

Copyright © 2011 Microsoft Corporation.

The exception to this is the use of the Netlogon RPC interface [MS-NRPC]. There, the purpose of the interface is to provide an ongoing connection to the domain controller for the purpose of authentication and authorization. The domain client will establish a connection over Netlogon RPC when necessary (typically, the first time it needs to authenticate a connection), and keep that connection so long as the client and the domain controller server are both running.

4.8 Common Failure Scenarios

The Tasks in this document specify a series of actions that effect the necessary state changes such that the client is joined to the domain. These changes include those that are local to the client, and those that occur in the domain (that is, those that create or modify a computer account object on a domain controller (DC)). In general, failure of any one particular action must cause failure of the Task. Exceptions to this principle are specified where necessary (for example, failed updates to the [MS-SNTP] protocol during join or unjoin processing are ignored).

While unlikely, the Tasks in this document may fail when making local (client) state changes. Such failures may occur due to reasons such as resource starvation. The Tasks in this document do not attempt to remedy these failure conditions; the only recourse is for the Task caller to re-execute the Task.

When communicating with a remote machine such as a domain controller, some obvious potential failure conditions include lack of network connectivity, or insufficient security privileges to create or modify a computer account object. The Tasks in this document do not attempt to remedy these failure conditions; the only recourse is for the Task caller to re-execute the task. When a Task is re-executed, no assumptions should be made about the state of a computer account object in the domain.

All Tasks make reasonable efforts in the face of failure to restore local client state to the original starting state. If those efforts fail, administrator intervention (outside the scope of the Task) may be necessary. Similarly, if a Task successfully creates or modifies a computer account object in the domain but then fails in a later step, the Task will make reasonable efforts to either disable or delete the computer account object. Failure to disable or delete the computer account object in that case may require domain administrator intervention (outside the scope of the Task) to make the changes manually.

Copyright © 2011 Microsoft Corporation.

5 Locating a Domain Controller

This section describes the Locating a Domain Controller task. This task is used by a client in order to access resources in the domain.

Locating a domain controller is the first step in the process of a client joining a domain. Therefore, all subsequent tasks defined in this document depend on this task.

5.1 Task Overview

5.1.1 Task Purpose

The purpose of the Locating a Domain Controller task is to enable a client to locate a domain controller within a domain or forest using a DNS or NetBIOS domain name.

5.1.2 Task Applicability

This task is applicable to clients that need to locate a domain controller (DC) in order to access resources in the domain.

5.1.3 Task Use Cases

The following sections describe the stakeholders and interest summaries.

5.1.3.1 Stakeholders and Interests Summary

Client User: The end user of the system accessing resources within the domain.

Client Application: An application running on a Client Computer that wants to access resources within the domain.

Client Computer: A computer that needs to locate a domain controller in order to access domain resources.

Domain Controller: The domain controller is a computer providing domain services to domain clients. In order for a domain controller to serve clients, there must be a mechanism for the clients to locate it.

NetBIOS Infrastructure: An infrastructure that supports NetBIOS name resolution ([RFC1001], [RFC1002]).

DNS Infrastructure: An infrastructure that provides DNS name resolution.

This task relies on at least one of the previously listed infrastructures to be available to discover domain controllers that could satisfy the requested capabilities.

5.1.3.2 Supporting Actors and Task Interests Summary

This task is primarily invoked by the **DsrGetDcName**, **DsrGetDcNameEx**, and **DsrGetDcNameEx2** methods of the Netlogon Remote Protocol ([MS-NRPC] sections 3.5.5.3.1, 3.5.5.3.2, and 3.5.5.3.3). Domain clients use one of these methods to locate a domain controller in order to access resources in the domain.

Copyright © 2011 Microsoft Corporation.

In addition, this task is invoked directly by the tasks specified in sections 6, 7, and 8 (where the client is not yet joined to the domain and the Netlogon Remote Protocol is therefore not initialized) to locate a domain controller.

5.1.3.3 Use Case Diagrams

The following diagram shows the use case of a client attempting to locate a domain controller:

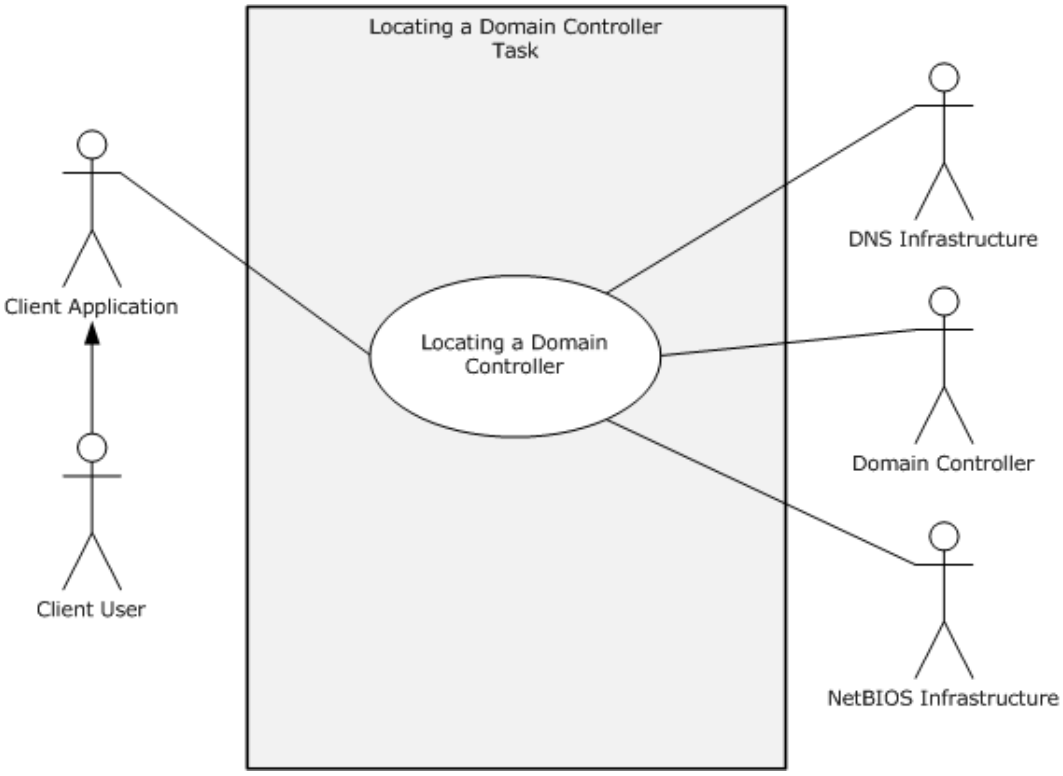

Figure 16: Locating a domain controller - domain client

5.1.3.4 Locating a Domain Controller — Client Application

Goal: Locate a domain controller (DC) in order to perform domain-oriented actions as illustrated in the use case figure.

Context of use: When a Client Application needs to access resources in a domain, locating a DC is the first step in the process.

Direct Actor: The direct actor is the Client Application that wants to access resources within the domain.

Primary Actor: The primary actor is the Client User.

Supporting Actors: NetBIOS Infrastructure and DNS Infrastructure, as specified in section 5.1.3.1.

Stakeholders and Interests: All stakeholders specified in section 5.1.3.1.

Minimal Guarantees: Either a domain controller that meets the criteria specified via the task abstract parameters is located and information about the domain controller is provided to the caller

Copyright © 2011 Microsoft Corporation.

as described in the task abstract results section (section 5.3.4), or an error is returned indicating that a domain controller could not be located.

Upon failure, the local Client Computer state defined in section 4.3.1.1 is unchanged.

Success Guarantee: A domain controller that meets the criteria specified by the task abstract parameters is located, the task abstract results (see Section 5.3.4) are returned to the caller with information about the domain controller.

Main Success Scenario:

1. The FQDN(2) of the domain, in which the domain controller is to be located, is available.
2. The client computer uses the FQDN(2) domain name to query the DNS infrastructure for relevant SRV records.
3. The client computer receives one or more SRV records that match the criteria from the DNS server.
4. The client computer resolves the name of the domain controllers using DNS infrastructure to get the IP addresses, and contacts the domain controllers via an LDAP Ping ([MS-ADTS] section 7.3.3) to determine "liveness" and confirm that the requested capabilities are present.
5. At least one domain controller that satisfies the client's requirements responds to the client computer's ping.
6. Information about the domain controller is returned to the initiator of the task.

Extensions/Variations:

If only the NetBIOS domain name of the domain is available, then the client computer contacts the candidate domain controllers via a MAILSLOT ping ([MS-ADTS] section 7.3.5) sent to a NetBIOS group name ([MS-MAIL] section 3.1.4) that is registered by domain controllers ([MS-ADTS] section 7.3.5).

If at least one domain controller that satisfies the client's requirements responds to the client computer's ping, then information about any one of the appropriately responding domain controllers is returned to the initiator of the task.

Otherwise, a domain controller could not be located, and an error is returned to the task initiator indicating that a domain controller could not be located.

5.2 Task Context

This section describes the relationship between this task and its environment.

5.2.1 Task Environment

This task is useful in a domain environment where **client computers** need to locate a domain controller for the purpose of joining a domain, or performing other domain-related operations. The task requires that either a DNS infrastructure or a NetBIOS infrastructure ([RFC1001] and [RFC1002]) exists where the domain controllers register their information and client computers can query the information to be able to locate the domain controllers.

Copyright © 2011 Microsoft Corporation.

5.2.2 Task Relationships

This task is the first step toward a client joining a domain. All other tasks in this document are reliant on this task being completed successfully.

5.2.2.1 Black Box Relationship Diagrams

The following diagram illustrates the client's process in locating a domain to join. Assumptions are that the client knows the name of the domain and that the domain is available to be located. The client uses the Domain Name System (DNS) to locate the desired domain.

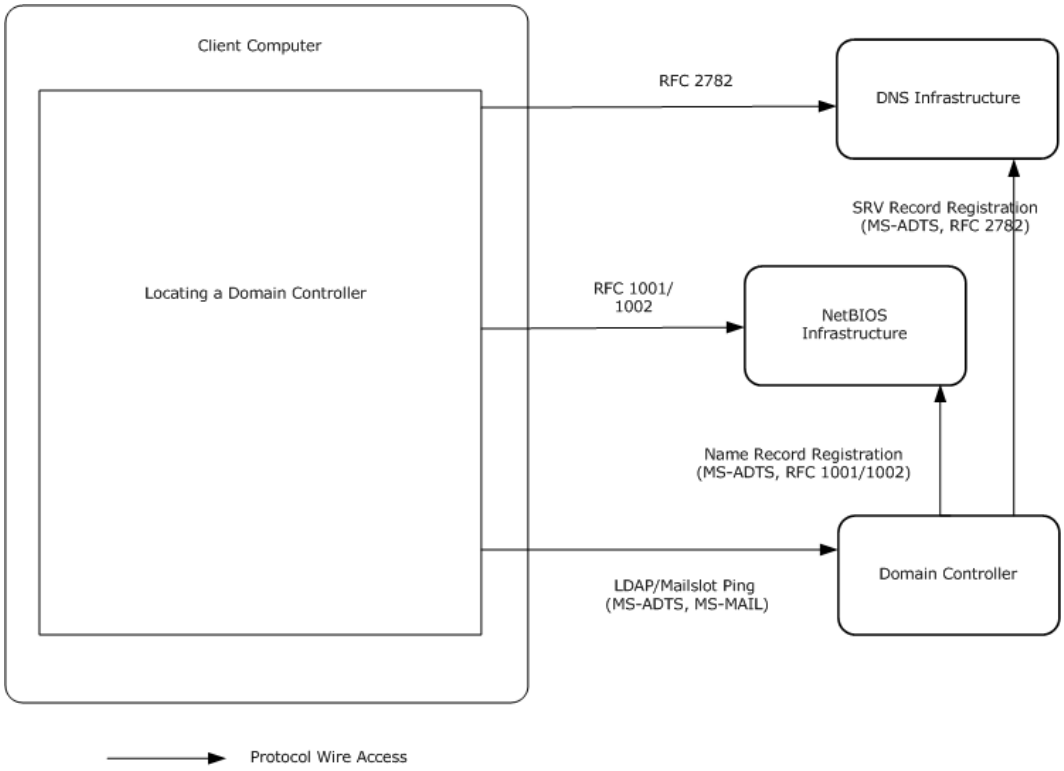

Figure 17: Locating a domain controller – black box relationships

5.2.2.2 Task Dependencies

Success of this task relies on domain controllers being present and registering information about their capabilities with a DNS and/or NetBIOS infrastructure. Both the domain client and the domain controller rely upon the networking infrastructure for their communication. When this task is successful, the other tasks included in this document can proceed.

5.2.2.3 Task Influences

None.

5.2.3 Task Assumptions and Preconditions

This task assumes the following:

Copyright © 2011 Microsoft Corporation.

- The domain client assumes basic network connectivity and the availability and necessary configurations to enable usage of basic network infrastructure services such as a DNS infrastructure and/or a NetBIOS Infrastructure.
- If a domain controller responds to the LDAP/Mailslot ping, the domain controller is considered "ready" for the capabilities that were listed in the ping response.

The following conditions are necessary for the success of the task:

- The domain controller publishes its name and capabilities with a DNS infrastructure and/or a NetBIOS infrastructure ([MS-ADTS] sections 7.3.2 and 7.3.4).

5.2.4 Task Versioning and Capability Negotiation

None.

5.3 Task Architecture

5.3.1 Task Architectural Constraints

This task is triggered by other tasks. Multiple instances of this task can run at the same time.

5.3.2 Task Abstract Data Model

This section describes state established, used, and maintained by processing rules of this task. State may be volatile or persisted. State may pertain to one, some, or all instances of the task. The task's state consists of the values of the named data elements (also called state variables) presented in this section. The overall organization of the data elements, with their names, is the Abstract Data Model. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules may depend upon associations established by the structure of its Abstract Data Model, such association can be achieved in other ways. Implementations may depart from this model so long as their external behavior remains consistent with that described in this document.

The following ADM elements from the common client data model (section 4.3.1.1) are accessed in a read-only manner by this task: **TrustedDomains**, **ComputerName**, **DomainSid**.

The following ADM element from the common client data model is read, and in some cases updated, by this task: **SiteName**.

The following are the in-memory values used by the task. These values are not persisted.

Name	Type	Description
TaskLocalDnsDomainName	string (Unicode)	Contains the FQDN(2) of the domain in which the domain controller is to be located. This element is initialized to NULL.
TaskLocalNetBIOSDomainName	string (Unicode)	Contains the NetBIOS name of the domain in which the domain controller is to be located. This element is initialized to NULL.

5.3.3 Task Abstract Parameters

This section describes data passed to an instance of this task at the time it is invoked or triggered. The parameters consist of the values of the named data elements presented in this section. The

Copyright © 2011 Microsoft Corporation.

organization of a data element, with its names, is an Abstract Parameter. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules may depend upon associations established by the structure of its Abstract Parameters, such association can be achieved in other ways. Implementations can depart from this abstraction so long as their external behavior remains consistent with that described in this document.

The parameters to this task are as follows:

Name	Type	Description	Optional
TaskInputDomainName	string (Unicode)	Specifies the NetBIOS or FQDN name of the domain for which to locate a domain controller.	Yes
TaskInputCapabilities	integer	Specifies the desired characteristics of the DC, as specified in DS_FLAGS_OPTIONS (see [MS-ADTS] section 7.3.1.2).	No
TaskInputAccountName	string (Unicode)	Specifies a user or machine account name, which the DC located by this task MUST have knowledge of.	Yes
TaskInputAccountControlBits	string (Unicode)	Specifies account control bits (see **userAccountControl** attribute, [MS-ADTS] section 2.2.15) of **TaskInputAccountName**. This parameter MUST be specified if **TaskInputAccountName** is specified.	Yes

5.3.4 Task Abstract Results

This section describes data returned by an instance of this task to its caller. The results consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Result. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules may depend upon associations established by the structure of its Abstract Results, such association can be achieved in other ways. Implementations can depart from this abstraction so long as their external behavior remains consistent with that described in this document.

The task returns the following results to the caller:

Name	Type	Description
TaskReturnStatus	integer	This task MUST return 0x00000000 on success. Error statuses generated by a failure during task processing are in the Win32 error space (a long data type), as specified in [MS-ERREF] section 2.2.
TaskReturnDomainController.Name	string	Upon successful completion of the task, returns the name of the selected domain controller. The name can be either the FQDN(1) or the NetBIOS name.
TaskReturnDomainController.Address	string	Upon successful completion of the task,

Copyright © 2011 Microsoft Corporation.

Name	Type	Description
		returns the IP address (if available) or the NetBIOS name of the selected domain controller.
TaskReturnDomainController.AddressType	enum (IPAddress,NetBIOS)	Upon successful completion of the task, returns the type of address returned in the **TaskReturnDomainController.Address** field.

5.3.5 White-Box Relationships

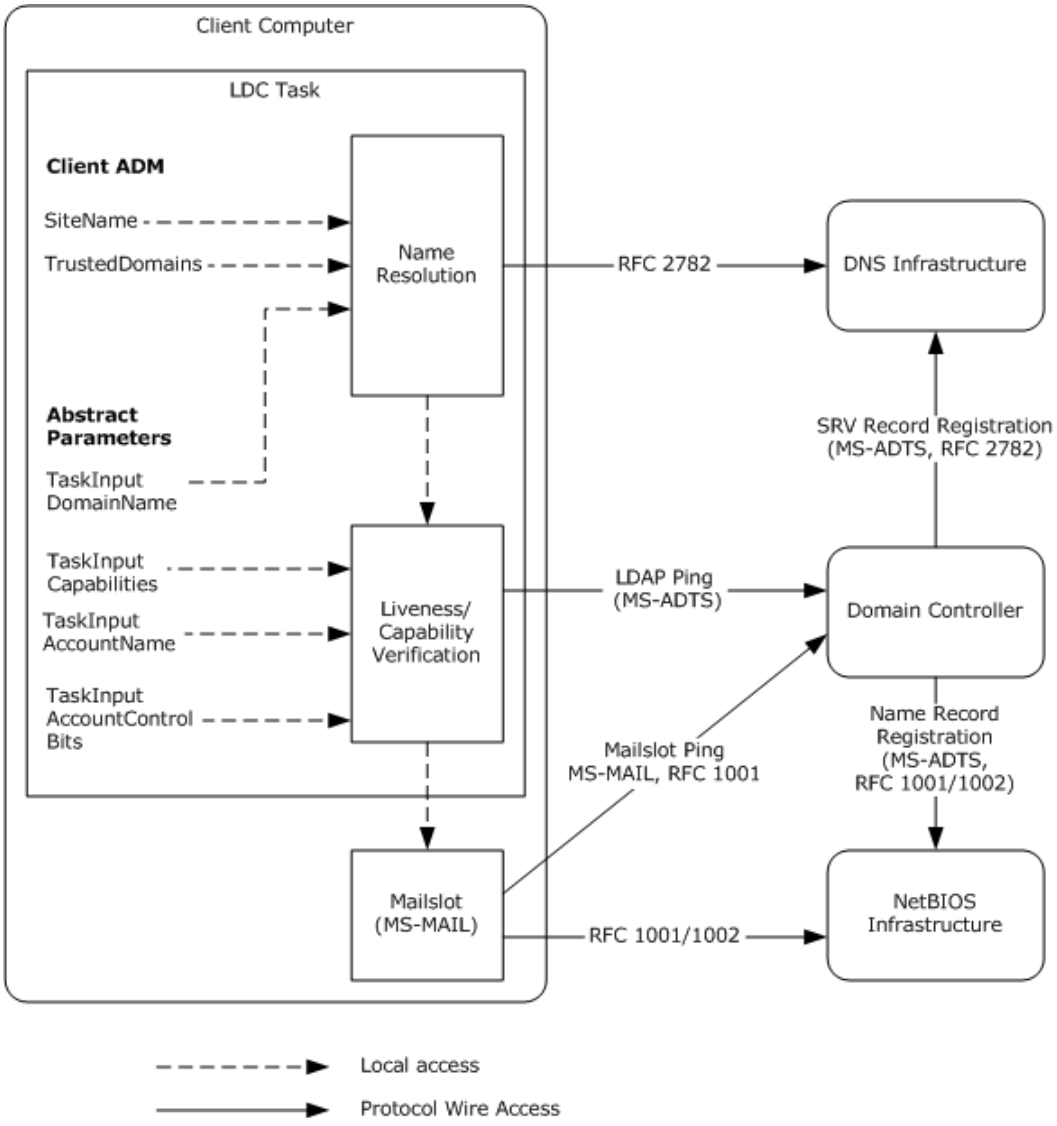

Figure 18: Locating a domain controller – white-box relationships

Copyright © 2011 Microsoft Corporation.

The preceding figure represents the relationships in the Locating a Domain Controller Task component.

The two main subcomponents represented are: Name Resolution and Liveness/Capability Verification, which are detailed further in section 5.3.7.

The Abstract Data Model (ADM) state elements present here (**SiteName**, **TrustedDomains**) are detailed further in section 5.3.2. The Abstract Parameters present here (**TaskInputDomainName**, **TaskInputCapabilities**, **TaskInputAccountName**, and **TaskInputAccountControlBits**) are detailed further in section 5.3.3.

The main success scenario and extensions are detailed in section 5.3.8, while the failure scenarios are described in section 5.3.9.

Refer to section 5.4.4 for the architectural details, and more specifically refer to 5.4.4.1 for the DNS approach, and section 5.4.4.2 for the NetBios approach. Further details about the rule details, interactions between the different components in the diagram, as well as the processing sequence represented as a flowchart, are discussed in section 5.4.5.

5.3.6 Task Events

5.3.6.1 Task Timers

None.

5.3.6.2 Task Non-Timer Events

None.

5.3.7 Task Architecture and Communication

This task consists of two subcomponents (as shown in the following figure): name resolution, and liveness/capability verification.

Name Resolution: This subcomponent is responsible for identifying a list of candidate DCs based on the specified abstract parameters and the client's ADM. The name resolution is done by querying the DNS infrastructure.

Liveness/Capability Verification: This subcomponent is responsible for verifying that a candidate domain controller is reachable and satisfies the specified requirements (through abstract parameters to this task). The verification is done by means of the LDAP Ping (as described in [MS-ADTS] section 7.3.3) or the MAILSLOT ping (as described in [MS-ADTS] section 7.3.5).

The following figure shows the two sub-components of the task and how each subcomponent interacts with external entities within the system.

Copyright © 2011 Microsoft Corporation.

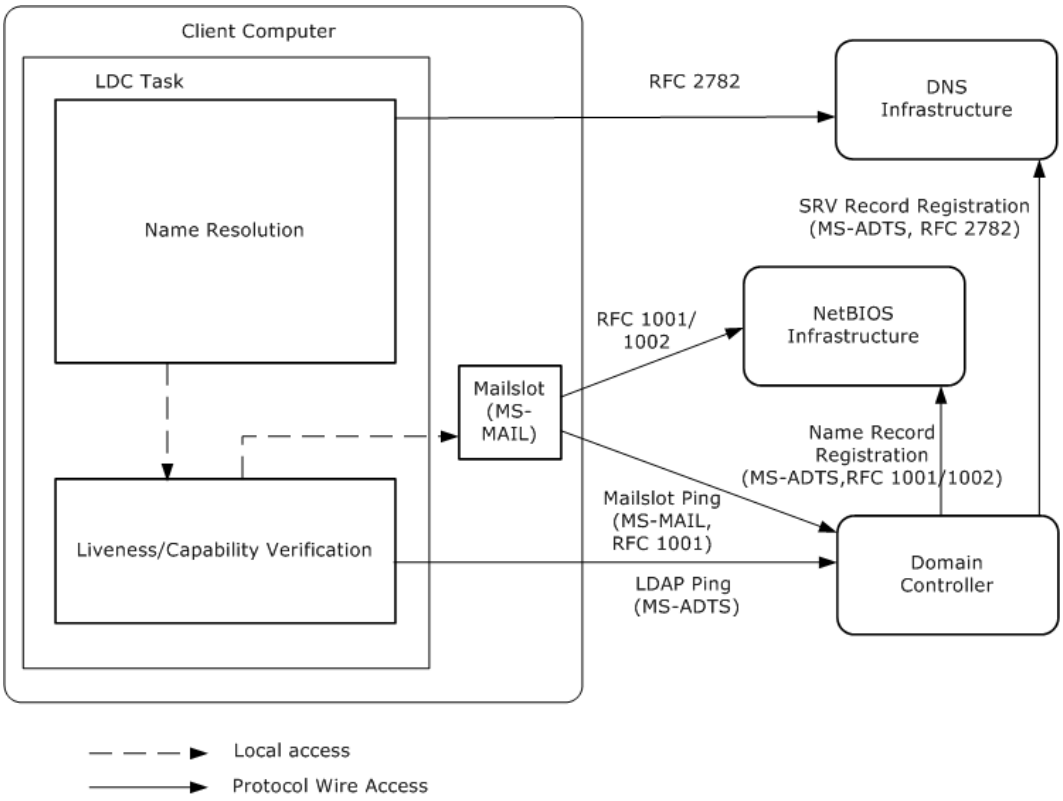

Figure 19: Locating a domain controller – task architecture

5.3.8 Task Processing Rules

This section outlines the task's high-level steps. See section 5.4.5 for the details on each step.

Task: Locating a Domain Controller.

Abstract Parameters: As specified in section 5.3.3.

Preconditions: See section 5.2.3.

Main Success Scenario:

1. Determine the FQDN(2) of the domain in which the domain controller is to be located based on the abstract parameter **TaskInputDomainName**. See section 5.4.5.1 for details.
2. Use the FQDN(2) of the domain (obtained in step 1) and site information from the client's ADM to get a list of IP addresses of candidate domain controllers using DNS infrastructure. See section 5.4.5.2 for details.
3. Contact each candidate domain controller via an LDAP Ping until a domain controller is located that supports all the specified requirements (specified via abstract parameters **TaskInputCapabilities**, **TaskInputAccountName**, and **TaskInputAccountControlBits**). See section 5.4.5.3 for details.
4. Information about the domain controller is returned to the initiator of the task. See section 5.4.5.6 for details.

Copyright © 2011 Microsoft Corporation.

Extensions:

1.

1.a. The abstract parameter Domain is not a FQDN(2), and the corresponding FQDN(2) of the domain cannot be determined from it. Clients MUST perform the following steps to attempt a NetBIOS-based domain controller discovery.

1.b. Determine the NetBIOS name of the domain based on the abstract parameter **TaskInputDomainName** and the client's ADM. If the NetBIOS name cannot be determined, the task returns an error to the task initiator, indicating that a domain controller could not be located. See section 5.4.5.4 for details.

1.c. Use the Remote Mailslot Protocol ([MS-MAIL]) to send a MAILSLOT ping to candidate domain controllers. The ping response is used to determine availability and to confirm that the domain controller supports all the specified requirements (specified via abstract parameters **TaskInputCapabilities**, **TaskInputAccountName**, and **TaskInputAccountControlBits**).

If no domain controllers respond or none of the domain controllers support all the requirements, the task returns an error to the task initiator indicating that a domain controller could not be located. See section 5.4.5.5 for details.

1.d. Information about the domain controller is returned to the initiator of the task. See section 5.4.5.6 for details.

2.

2.a. If the DNS infrastructure cannot be contacted, go to 1.b.

2.b. If no candidate domain controllers are returned by the DNS infrastructure, go to 1.b.

3.

3.a. If each candidate domain controller has been contacted and either no domain controller is available (no LDAP ping response) or the available domain controllers do not support all the specified requirements, go to 1.b.

5.3.9 Task Failure Scenarios

There are two primary failure scenarios for this task:

Network Infrastructure Unavailable: In this situation, neither DNS nor NetBIOS infrastructure is available (for example, extension 2a in the preceding section). The task is unable to identify the candidate domain controllers and the task is terminated with an error.

Domain Controller Unavailable: In this situation, either no domain controllers are located through DNS name resolution (illustrated as extension 2b in the preceding section), or no domain controller responds to the LDAP/MAILSLOT ping, or the domain controllers that do respond do not satisfy all the specified requirements (illustrated as extensions 1a1b1 and 3a). In this case, the task is terminated with an error.

5.4 Task Details

This section contains the details that complete the descriptions in earlier sections of the document. These are needed to understand and implement this task.

Copyright © 2011 Microsoft Corporation.

5.4.1 Task Precondition Details

Not applicable.

5.4.2 Task Initialization of External Entities

None.

5.4.3 Task Event Details

5.4.3.1 Task Timer Details

None.

5.4.3.2 Task Non-Timer Event Details

None.

5.4.4 Task Architectural Details

This task can be performed by using one or both of two possible approaches: DNS-based location and NetBIOS-based location. If the FQDN(2) of the domain is available, the DNS-based location MUST be performed. If only the NetBIOS name of the domain is available, or if the DNS-based location is unsuccessful, the NetBIOS-based location MUST be performed.

A high level overview of each location approach is presented in the following sections. Details are provided in section 5.4.5.

5.4.4.1 Location Based on DNS Domain Name

As described in [MS-ADTS] section 7.3.2, all domain controllers register SRV records in DNS. The DNS-based location approach leverages these SRV records to obtain a list of candidate domain controllers that can potentially satisfy the requirements of the client. Each domain controller in this list is then contacted via an LDAP ping until one that is available and that meets all specified requirements is found.<4><5><6>The following figure shows the sequence of interactions that the client has with the DNS infrastructure and the domain controller(s) as part of this location mechanism.

Copyright © 2011 Microsoft Corporation.

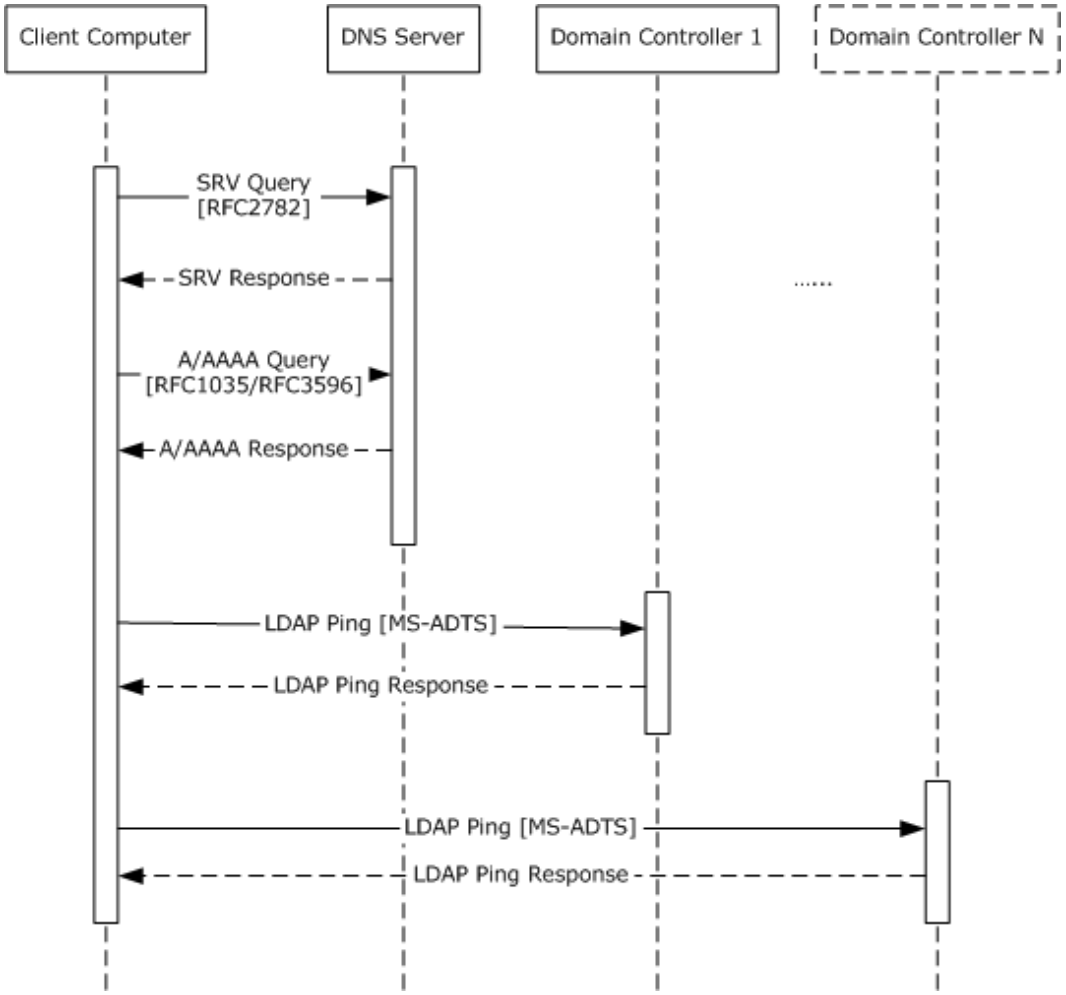

Figure 20: Location based on DNS domain name

5.4.4.2 Location Based on NetBIOS Domain Name

Locating a domain controller based on the NetBIOS domain name relies on NetBIOS group names, which can be implemented either by broadcast or by querying a NetBIOS Name Server (NBNS). A working NetBIOS infrastructure is required for this locator method to work effectively. NetBIOS over IP is covered in [RFC1001] and [RFC1002]. As described in [MS-ADTS] section 7.3.4, all domain controllers register certain NetBIOS group names with the NetBIOS infrastructure. In the NetBIOS based location approach, the client sends a mailslot ping to candidate domain controllers by targeting the appropriate group names via the Remote Mailslot Protocol ([MS-MAIL]). The following figure shows the sequence of interactions that the client has with the NetBIOS infrastructure and the domain controller as part of this location mechanism.

Copyright © 2011 Microsoft Corporation.

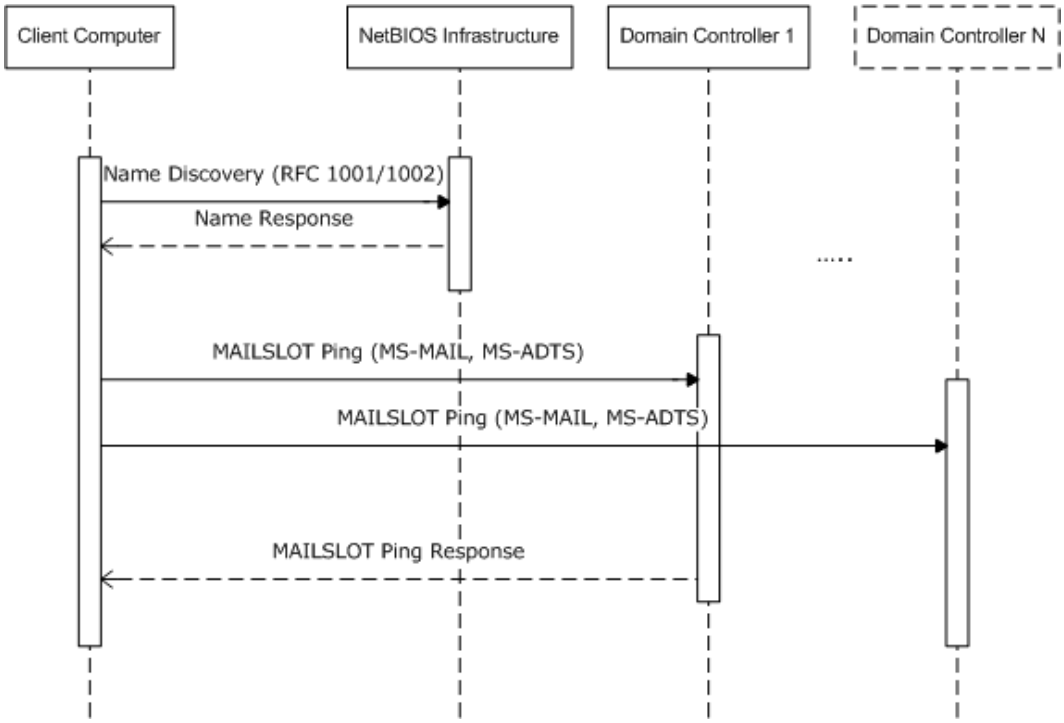

Figure 21: Location based on NetBIOS domain name (via NBNS and Broadcast)

5.4.5 Task Processing Rule Details

This section describes details for the steps identified in section 5.3.8. Unless otherwise specified, the processing falls through from one section to the next. The following figure captures the processing sequence as a flowchart.

Copyright © 2011 Microsoft Corporation.

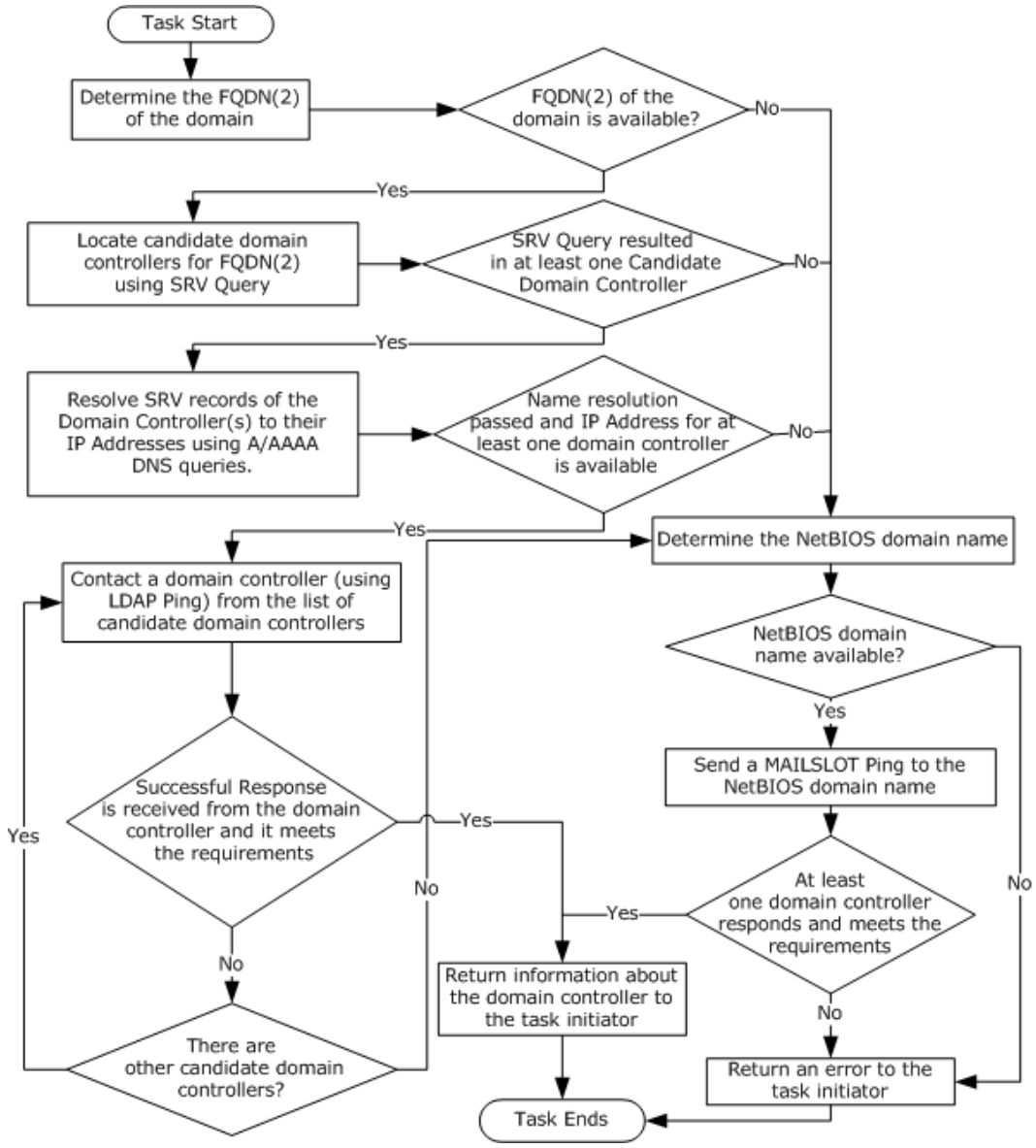

Figure 22: Processing sequence

5.4.5.1 Determine DNS Domain Name of the Domain

TaskLocalDnsDomainName MUST be initialized as follows:

1. If the abstract parameter **TaskInputDomainName** is specified, this name is looked up in the client's ADM element **TrustedDomains**. If a match is found, and the matching tuple's FQDN is set, **TaskLocalDnsDomainName** is initialized with the FQDN(2) the matching tuple.

2. If the abstract parameter **TaskInputDomainName** is specified, but no match is found in the **TrustedDomains** and **TaskInputDomainName** appears to be a syntactically valid FQDN(2), **TaskLocalDnsDomainName** is initialized to **TaskInputDomainName**.

Copyright © 2011 Microsoft Corporation.

3. If the abstract parameter **TaskInputDomain** is not specified (is NULL), then **TaskLocalDnsDomainName** is initialized to the value of **DomainName.FQDN** as specified in section 4.3.1.1. If **DomainName.FQDN** is NULL, meaning the client is not currently joined to a domain, the task MUST return an error.

If **TaskLocalDnsDomainName** cannot be initialized using any of the preceding rules, DNS based location is not possible. The task MUST fallback to the NetBIOS-based location as described in sections 5.4.5.4 and 5.4.5.5.

5.4.5.2 Identify List of Candidate Domain Controllers Based on DNS Information

After **TaskLocalDnsDomainName** is determined, a list of candidate domain controllers is obtained by querying **DNS** for **SRV** records. The client MUST construct the SRV query as follows:

1. Based on the value of the flags **DS_PDC_FLAG**, **DS_LDAP_FLAG**, **DS_GC_FLAG**, **DS_KDC_FLAG**, and **DS_DS_FLAG** of the abstract parameter **TaskInputCapabilities** (which lists the requested capabilities of the domain controller), the appropriate query is selected from those listed in [MS-ADTS] section 7.3.6. Other capabilities specified in **TaskInputCapabilities** do not contribute to the selection of this query. Capabilities specified in **TaskInputCapabilities** are used to validate the capabilities in the response (see sections 5.4.5.3 and 5.4.5.5).

2. If the client's **SiteName** ADM element is initialized, the client MUST attempt to first use a site-specific query where applicable. For example, if the client is attempting to locate a KDC, the following query could be used: _kerberos._tcp.<**SiteName**>._sites.dc._msdcs.<**TaskLocalDnsDomainName**>.

3. If the client's **SiteName** ADM element is not initialized, or the site-specific query does not result in any candidate domain controllers, or if the candidate domain controllers are not reachable via LDAP Ping (described in section 5.4.5.3), clients MUST fallback to non site-specific queries. Using the same KDC example as before, the following query could be used: _kerberos._tcp.dc._mcdcs.<**TaskLocalDnsDomainName**>.

The DNS exchange is done as specified in the DNS protocols (RFC2782 and related RFCs).

If no candidate domain controllers can be identified using SRV queries, the client MUST fallback to NetBIOS based location as described in sections 5.4.5.4 and 5.4.5.5.

5.4.5.3 Ping the Candidate Domain Controllers for "Liveness" and Capability Verification Using LDAP Ping Mechanism

Once the list of candidate domain controllers is obtained using the DNS infrastructure, the client MUST select a candidate domain controller based on weighted random order (see [RFC2052]). The client then resolves the SRV record to an IP address using A/AAAA DNS queries.

Once the address is known, the client sends an LDAP "Ping", [MS-ADTS] section 7.3.3, to the candidate domain controller to determine whether the domain controller is in fact handling requests and whether its capabilities satisfy the client requirements.

The LDAP Ping request MUST be constructed as an LDAP rootDSE search with an LDAP filter that is a one-level AND of equalityMatch tests of the following elements:

DnsDomain: Equals **TaskLocalDnsDomainName**.

Host: Equals **ComputerName.NetBIOS** from the client's ADM.

Copyright © 2011 Microsoft Corporation.

User: Equals **TaskInputAccountName** if the parameter **TaskInputAccountName** is specified, else this element is not included in the filter.

AAC: Equals **TaskInputAccountControlBits** if the parameter **TaskInputAccountControlBits** is specified, else this element is not included in the filter.

NtVersion: NETLOGON_NT_VERSION options (see [MS-ADTS] section 7.3.1.1). This MUST be set as follows.

1. The following bits are always set: NETLOGON_NT_VERSION_5, NETLOGON_NT_VERSION_5EX, and NETLOGON_NT_VERSION_WITH_CLOSEST_SITE.<7>

2. In addition, other bits are set depending on the value of the abstract parameter **TaskInputCapabilities** as follows:

Flag in TaskInputCapabilities	NETLOGON_NT_VERSION Bit
DS_GC_FLAG	NETLOGON_NT_VERSION_GC
DS_PDC_FLAG	NETLOGON_NT_VERSION_PDC

The client sends the LDAP Ping request as follows:

1. The client invokes the task **Initializing an ADUDPHandle** ([MS-ADSO] section 6.2.7.3.1) with the following parameters:

   - **TaskInputTargetName** is set to the address of the domain controller.
   - **TaskInputPortNumber** is set to 389.
   - **TaskInputProtocolVersion** is set to 3.

2. Step 1 above returns an LDAP_UDP_HANDLE which is a pointer to an instance of **ADUDPHandle** described in [MS-ADSO] section 6.2.7.2 (ADUDPHandle Abstract Data Model). Let the returned handle be **adUDPHandle**.

3. Let **ldapPingResultMessages** be a list of **LDAPMessage** objects.

4. The client invokes the task **Performing an LDAP operation on an ADUDPHandle** ([MS-ADSO] section 6.2.7.3.2) with the following parameters:

   - **TaskInputADUDPHandle** is set to **adUDPHandle**.
   - **TaskInputRequestMessage** is set to the LDAP Ping request constructed above.
   - **TaskOutputResultMessages** is set to **ldapPingResultMessages**.
   - **TaskInputRequestTimeout** is set according to the timeout/retry logic that the client implements (The behavior of a Microsoft Windows® domain client is described below).

5. Step 4 above returns **taskReturnStatus** indicating the result of the LDAP operation.

If there is no response, that is, if **taskReturnStatus** in step 4 above indicates that a timeout has occurred, the client SHOULD implement some level of retry logic, selecting other entries from the SRV request response. If the client retries too much, then responsiveness on the client can be reduced. If the client retries too few times, then there is the risk of not finding a domain controller because of transient network conditions.<8>

Copyright © 2011 Microsoft Corporation.

Upon receipt of a successful LDAP Ping response returned in **ldapPingResultMessages** in step 3 above, the client MUST validate that the capabilities returned by the domain controller satisfy the requested capabilities (abstract parameters **TaskInputCapabilities**). The capabilities are typically returned in a NETLOGON_SAM_LOGON_RESPONSE_EX structure ([MS-ADTS] section 7.3.1.9), but can be in a NETLOGON_SAM_LOGON_RESPONSE or NETLOGON_SAM_LOGON_RESPONSE_NT40 ([MS-ADTS] sections 7.3.1.8 and 7.3.1.7, respectively). The format of the response buffer is in response to the version information sent by the client in the NETLOGON_NT_VERSION element. This is fully detailed in [MS-ADTS] section 7.3.3.2.

If the capabilities returned by the domain controller are incompatible with the requirements specified by the invoker of the locator algorithm on entry, the client has to select another candidate domain controller from the list of domain controller SRV records returned above. Incompatibility in this case can arise because the client requested a Kerberos KDC, but the domain controller did not indicate that a KDC was present, or the client requested a domain controller that could accept writes, but this domain controller is read-only. This is not a catastrophic error; the client has to simply move on to another candidate domain controller.

Windows domain clients maintain a list of domain controllers as described in section 5.4.5.2. For the first five domain controllers selected from the list, a timeout value of 0.4 seconds is used. For the next five domain controllers selected from the list, a timeout value of 0.2 seconds is used and for the remaining domain controllers, a timeout value of 0.1 seconds is used. If the LDAP Ping request times out, or capabilities returned by the domain controller do not satisfy the requested capabilities, then that domain controller is considered not to be available.

If all the responses in the SRV records have been checked, and each SRV record points to a server that is either not available or does not match the requirements, then the location operation has failed. The client MUST fallback to NetBIOS based location as described in section 5.4.5.4 and section 5.4.5.5.

5.4.5.4 Determine NetBIOS Name of the Domain

Let **TaskLocalNetBIOSDomainName** be the NetBIOS name of the domain in which a domain controller is to be located. **TaskLocalNetBIOSDomainName** MUST be initialized as follows:

1. If the abstract parameter **TaskInputDomainName** is specified, this name is looked up in the client's ADM element **TrustedDomains**. If a match is found, and the matching tuple's NetBIOS name is set, **TaskLocalNetBIOSDomainName** is initialized with it.

2. If the abstract parameter **TaskInputDomainName** is specified, but no match is found in the **TrustedDomains** client ADM and **TaskInputDomainName** appears to be a syntactically valid NetBIOS name, **TaskLocalNetBIOSDomainName** is initialized to **TaskInputDomainName**.

If **TaskLocalNetBIOSDomainName** cannot be initialized using any of the above rules, NetBIOS based location is not possible. The task MUST return an error to the task initiator indicating that a domain controller could not be located.

5.4.5.5 Location of Domain Controllers Based on NetBIOS Group Names

The client uses NetBIOS group names to locate operational DCs. See [MS-ADTS] section 7.3.4 and [MS-MAIL] section 3.2.3 for the group names that are registered by domain controllers and their associated capabilities. All domain controllers register a mailslot name of "\\mailslot\net\netlogon" ([MS-ADTS] section 7.3.5)

The following mailslot message (see [MS-MAIL] section 3.1.4.1) MUST be sent to locate a domain controller:

Copyright © 2011 Microsoft Corporation.

1. MailslotName = "\\mailslot\net\netlogon"

2. Based on the **TaskInputCapabilities** abstract parameter, if a primary domain controller is required, the TargetName is set to < NetBIOSDomainName >[1B], else it is set to < NetBIOSDomainName >[1C].

3. Based on the **TaskInputCapabilities** abstract parameter if a primary domain controller is required, the Message is initialized as a NETLOGON_LOGON_QUERY structure ([MS-ADTS] section 7.3.1.4), else as a NETLOGON_SAM_NETLOGON_REQUEST structure ([MS-ADTS] section 7.3.1.6). See the following details about how each of these structures is initialized based on the abstract parameters.

   The NETLOGON_LOGON_QUERY structure MUST be constructed with the following fields initialized as follows:

   Opcode: Set to LOGON_PRIMARY_QUERY operation code (see section 7.3.1.3).

   ComputerName: Set to **ComputerName.NetBIOS** from the client's ADM.

   UnicodeComputerName: Set to **ComputerName.NetBIOS** from the client's ADM.

   NtVersion: NETLOGON_NT_VERSION options (see [MS-ADTS] section 7.3.1.1). The following bits are always set: NETLOGON_NT_VERSION_5, NETLOGON_NT_VERSION_5EX_WITH_IP, and NETLOGON_NT_VERSION_PDC.

   LmNtToken (2 bytes): This MUST be set to 0xFFFF.

   Lm20Token (2 bytes): This MUST be set to 0xFFFF.

   The NETLOGON_SAM_NETLOGON_REQUEST MUST be constructed with the following fields initialized as follows:

   Opcode (2 bytes): Set to LOGON_SAM_LOGON_REQUEST operation code (see section 7.3.1.3).

   UnicodeComputerName: Set to **ComputerName.NetBIOS** from the client's ADM.

   UnicodeUserName: If the abstract parameter **TaskInputAccountName** was specified, it is set to that value, else NULL.

   AllowableAccountControlBits: If the abstract parameter **TaskInputAccountControlBits** was specified, it is set to that value, else 0.

   DomainSidSize (4 bytes): A DWORD that contains the size of the **DomainSid** field.

   DomainSid (variable): Set to **DomainSid** from the client's ADM.

   NtVersion: NETLOGON_NT_VERSION options (see [MS-ADTS] section 7.3.1.1). This MUST be set as follows.

   1. The following bits are always set: NETLOGON_NT_VERSION_5, NETLOGON_NT_VERSION_5EX_WITH_IP.

Copyright © 2011 Microsoft Corporation.

2. If the DS_GC_FLAG bit is set in the abstract parameter **TaskInputCapabilities**, the NETLOGON_NT_VERSION_GC bit is set.

LmNtToken (2 bytes): This MUST be set to 0xFFFF.

Lm20Token (2 bytes): This MUST be set to 0xFFFF.

Upon receipt of a successful MAILSLOT Ping response, the client MUST validate that the capabilities returned by the domain controller satisfy the requested capabilities (abstract parameter **TaskInputCapabilities**). The capabilities are typically returned in a NETLOGON_SAM_LOGON_RESPONSE structure ([MS-ADTS] section 7.3.1.8) or a NETLOGON_PRIMARY_RESPONSE STRUCTURE ([MS-ADTS] section 7.3.1.5), but can also be NETLOGON_SAM_LOGON_RESPONSE_NT40 ([MS-ADTS] section 7.3.1.7). The format of the response buffer is in response to the version information sent by the client in the NETLOGON_NT_VERSION element. This is fully detailed in [MS-ADTS] section 7.3.3.2.

If no domain controllers respond or if none match the required capabilities, the client MUST return an error indicating that a domain controller could not be located.

5.4.5.6 Returning Results to the Task Initiator and Updating the Client ADM

If the task succeeds, the following occurs:

If the domain in which the domain controller is to be located is the same as the client computer's domain, the **SiteName** ADM element is updated with the client site name information returned as part of the LDAP/MAILSLOT ping response by the domain controller.

The abstract results are initialized as follows and returned to the task initiator.

- **TaskReturnStatusCode**: 0x00000000
- **TaskReturnDomainController.Name**: Set to either the FQDN(1) or the NetBIOS name of the domain controller that meets the specified requirements.<9>

  The FQDN(1) and the NetBIOS name of the domain controller are obtained from the LDAP/MAILSLOT Ping response.
- **TaskReturnDomainController.Address**: Set to the IP address of the domain controller, if available, otherwise set to the NetBIOS name of the domain controller.

  The IP address of the domain controller is obtained as follows:

  1. In the DNS based mechanism (sections 5.4.5.2 and 5.4.5.3), this is the IP address that was obtained from the A/AAAA DNS record and that was used to perform a successful LDAP ping.
  2. In the NetBIOS mechanism (sections 5.4.5.5), the IP address is obtained from the MAILSLOT ping response.
- **TaskReturnDomainController.AddressType**: Set to the enum value IPAddress, if the IP address of the domain controller is available, else it is set to the enum value NetBIOS.

If the task fails, the abstract results are initialized as follows and returned to the task initiator.

- **TaskReturnStatusCode**: a Win32 error code as specified in [MS-ERREF] section 2.2.
- **TaskReturnDomainController**: NULL.

Copyright © 2011 Microsoft Corporation.

5.5 Task Security

There are no task-specific security considerations. Please refer to the Security section of this specification and the Security sections of the referenced protocol Technical Documents.

Copyright © 2011 Microsoft Corporation.

6 Joining a Domain Using a Predefined Account

This section describes the process by which a client computer can join a domain using a predefined account configured in the domain. A predefined account needs to be configured with a password derived from the client computer name and hence this task is not secure by definition.

6.1 Task Overview

6.1.1 Task Purpose

The purpose of this task is to establish a client computer as a member of a domain using a predefined account in the domain.

6.1.2 Task Applicability

This task is applicable in the case where a predefined account needs to be used to join a client computer to a domain.

6.1.3 Task Use Cases

6.1.3.1 Stakeholders and Interests Summary

Client Administrator: The client administrator is the administrator of the client computer, interested in joining the client computer to the domain.

Client Computer: The client computer that is being joined to the domain.

Domain Administrator: The domain administrator is the administrator of the domain. The domain administrator configures the domain with a machine account that the current task uses to join the client to the domain. After the join is completed, the domain administrator is able to influence the client computer via other protocols; see sections 3.1 and 3.4, and [MS-WSO], for further details.

Domain Controller: The domain controller is a computer providing domain services to domain clients. The pre-defined machine account on the domain controller is modified by the client during task processing.

6.1.3.2 Supporting Actors and Task Interests Summary

This Task depends on the following supporting actors for the specified interests:

- SMB protocol ([MS-SMB2], [MS-SMB], or [MS-CIFS]), for opening/closing **SMB** sessions to a DC. The Task does not require that any specific SMB protocol be used.
- [MS-NRPC] protocol, for locating a domain controller (DC), establishing a **Netlogon binding** to a DC, for enumerating the **trusted domains**, and for updating the machine account password.
- [MS-LSAD] protocol, for retrieval of information about a domain from a DC.
- [MS-DRDM] protocol, for searching for the client computer account on a DC.

There are no other systems or tasks in which this task is an actor.

Copyright © 2011 Microsoft Corporation.

6.1.3.3 Use Case Diagrams

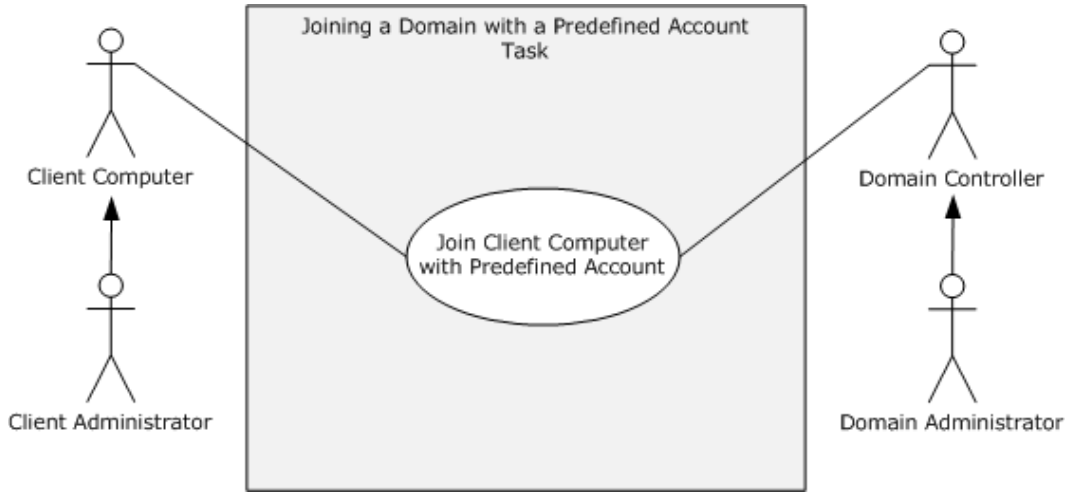

Figure 23: Joining a domain with a predefined account use case

6.1.3.4 Join a Client Computer to a Domain Using a Predefined Account — Client Computer

The only use case for this task is a client administrator joining the client computer to a domain using a predefined account on a domain controller.

Goal: Join a client computer to a domain using a predefined account.

Context of Use: This task is invoked by the client administrator in order to enable the client computer to access the services and resources in a domain as well as provide the domain members access to the client computer. See sections 3.1 and 3.4 for other motivations for joining a domain.

Direct Actor: The client administrator who wants to join the client computer to the domain.

Primary Actor: The client administrator who wants to join the client computer to the domain.

Direct Actor: The client computer joining the domain.

Supporting Actors: All supporting actors are specified in section 6.1.3.2.

Stakeholders and Interests: There are no other stakeholders for the current task besides those listed under primary actor, direct actor, and supporting actors sections as presented earlier.

Precondition: The preconditions for this use case are the same as those listed for the task in section 6.2.3.

Minimal Guarantees: This use case guarantees that upon failure, the local client computer state (section 4.3.1.1) is unchanged.

Success Guarantee: The client is joined to the domain.

Trigger: This use case is triggered by the client administrator to join the client to a domain.

Main Success Scenario:

Copyright © 2011 Microsoft Corporation.

1. The client administrator initiates the domain join task on the client computer.
2. The client locates a domain controller.
3. The client connects to the domain controller as the anonymous user (see [MS-NLMP] section 3.2.5.1.2) and retrieves domain information.
4. The client binds to the domain controller using predefined account credentials.
5. The client determines the trusted domains.
6. The client updates the local client state.
7. The client reinitializes local protocols.

Extensions: None.

6.2 Task Context

This section describes the relationship between this task and its environment.

6.2.1 Task Environment

The task requires the following environment:

- The client computer SHOULD have basic network connectivity and basic network infrastructure services like DNS SHOULD be available to the client and the domain controllers. This task will fail if this requirement is not met.
- The domain controller used for the join SHOULD be configured to accept anonymous SMB sessions.

6.2.2 Task Relationships

This task builds on the Common Task Information (see section 4), which is shared with all of the tasks in this document.

6.2.2.1 Black Box Relationship Diagram

The following black box diagram illustrates the system as the client computer joins the domain using a predefined account and interacts with the domain controllers that are offering services to the client.

Copyright © 2011 Microsoft Corporation.

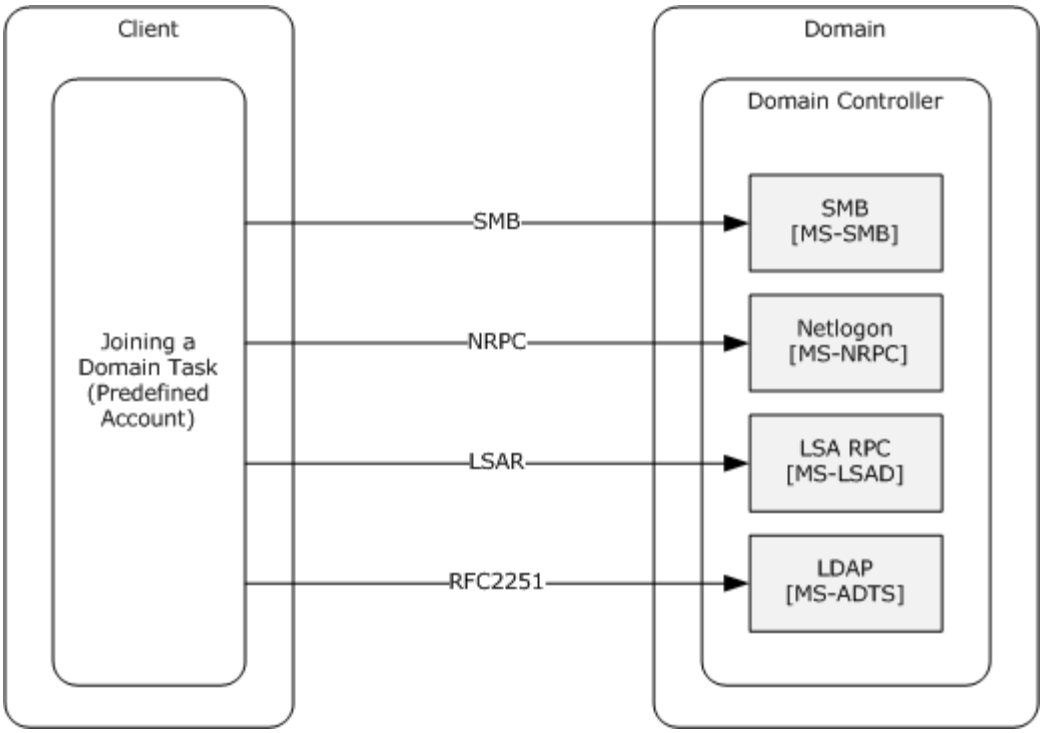

Figure 24: Client machine joining a domain using a predefined account

6.2.2.2 Task Dependencies

This task depends on the following:

- At least one domain controller being available.
- A domain controller being located.

6.2.2.3 Task Influences

None.

6.2.3 Task Assumptions and Preconditions

The following are the *success conditions* for this task:

- The administrator of the domain has created an account to represent the computer that wants to join the domain and this account is present on the domain controller selected by the client during task processing. The password for this account MUST be set to the machine's account name in lower case. This allows the client to have *a priori* knowledge of the key to use for authentication.
- The domain controller selected by the client is assumed to accept anonymous SMB sessions.

6.2.4 Task Versioning and Capability Negotiation

None.

Copyright © 2011 Microsoft Corporation.

6.3 Task Architecture

6.3.1 Task Architectural Constraints

This task has the following architectural constraints:

- The client computer can be joined to only a single domain at a time.
- Only one instance of this task MUST run on the client computer at any given time. As the task aims to join the client computer to a single domain, it does not support multiple, parallel executions.
- This task MAY be run a second time following a previous successful completion, in order to join the client computer to a different domain.
- This task MUST NOT make assumptions about distributed state, such as the machine account on the domain controller.
- This task MAY be run a second time following a previous *unsuccessful* run, as an unsuccessful run makes no changes to the client state.

6.3.2 Task Abstract Data Model

This section describes state established, used, and maintained by processing rules of this Task. State may be volatile or persisted. State may pertain to one, some, or all instances of the Task. The Task's state consists of the values of the named data elements (also called state variables) presented in this section. The overall organization of the data elements, with their names, is the Abstract Data Model. It is intended to facilitate the reader's conceptual understanding of the specification. While a Task's processing rules may depend upon associations established by the structure of its Abstract Data Model, such association can be achieved in other ways. Implementations may depart from this model so long as their external behavior remains consistent with that described in this document.

The following are the in-memory values used by the Task. These values are not persisted.

Name	Type	Description
TaskLocalClientName	string (Unicode)	Contains the samAccountName attribute value for the predefined account in the domain.
TaskLocalPassword	string (Unicode)	Contains the password for the predefined account in the domain.
TaskLocalNewPassword	string (Unicode)	Contains a randomly generated password.
TaskLocalDomainName.FQDN	string (Unicode)	Contains the FQDN(1) of the domain being joined.
TaskLocalDomainName.NetBIOS	string (Unicode)	Contains the NetBIOS name of the domain being joined.
TaskLocalDomainController	string (Unicode)	Contains the name of the domain controller used by this task.
TaskLocalForestNameFQDN	string (Unicode)	Contains the FQDN(1) of the forest containing the domain being joined.

Copyright © 2011 Microsoft Corporation.

Name	Type	Description
TaskLocalDomainSID	SID	Contains the SID of the domain being joined.
TaskLocalDomainGUID	GUID	Contains the GUID of the domain being joined.
TaskLocalTrustedDomains	Domains	Contains a list of trusted domains for the domain being joined.
TaskLocalClientAccountDN	string (Unicode)	Contains the DistinguishedName of the client account in the domain.
TaskLocalSMBSession	SMB/CIFS session	Contains the returned SMB state for the SMB/CIFS session established to the domain controller.
TaskLocalAlreadyJoined	Boolean	Whether the client is already joined to a domain at the start of the Task.

The following client data model variables (section 4.3.1.1) are updated by the Task during a successful completion:

- **DomainSid**
- **DomainGuid**
- **DomainName** (both NetBIOS and FQDN Names)
- **ClientName**
- **Password**
- **TrustedDomains**

6.3.3 Task Abstract Parameters

This section describes data passed to an instance of this task at the time it is invoked or triggered. The parameters consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Parameter. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules could depend upon associations established by the structure of its Abstract Parameters, such association can be achieved in other ways. Implementations can depart from this abstraction so long as their external behavior remains consistent with that described in this document.

The abstract parameters for this task are as follows:

Name	Type	Description	Optional
TaskInputDomainName	string (Unicode)	Specifies the NetBIOS or FQDN name of the domain to join.	No
TaskInputDomainController	string (Unicode)	Specifies the domain controller this task must use to complete the join.	Yes

6.3.4 Task Abstract Results

This section describes data returned by an instance of this task to its caller. The results consist of the values of the named data elements presented in this section. The organization of a data

Copyright © 2011 Microsoft Corporation.

element, with its names, is an Abstract Result. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules could depend upon associations established by the structure of its Abstract Results, such association can be achieved in other ways. Implementations can depart from this abstraction so long as their external behavior remains consistent with that described in this document.

The task returns the following results to the caller:

Name	Type	Description
TaskReturnStatus	integer	This task MUST return 0x00000000 on success. Error statuses generated by a failure during task processing are in the Win32 error space (a long data type), as specified in [MS-ERREF] section 2.2.

Upon a successful task completion, this task performs the following:

1. Updates client ADM variables (section 4.3.1.1). The updated values are persisted, as specified in section 4.3.1.1, and are available for other protocols on the client to use.
2. Starts the **NetLogon** Remote Protocol and configures it to run automatically at every system start up.
3. Notifies the Certificate Autoenrollment system that the machine is joining the domain.
4. Adds the Domain Administrators to the local Administrators group.
5. Notifies the local MS-SNTP protocol that the machine is joining the domain (failures in this step are ignored).

   Note that no changes are made to the pre-existing client computer account in the domain as a result of a successful task completion.

Upon a failed task completion, this task will not modify the state of the pre-existing computer account in the domain, and will attempt to revert any local state changes that were made prior to the task failure. If the task is unable to revert local state changes that were made, administrator intervention outside the scope of the task may be necessary to reconfigure the client back to its original state.

6.3.5 White-Box Relationships

The following diagram represents the white-box relationships of the task within the client computer and with the domain controller.

Copyright © 2011 Microsoft Corporation.

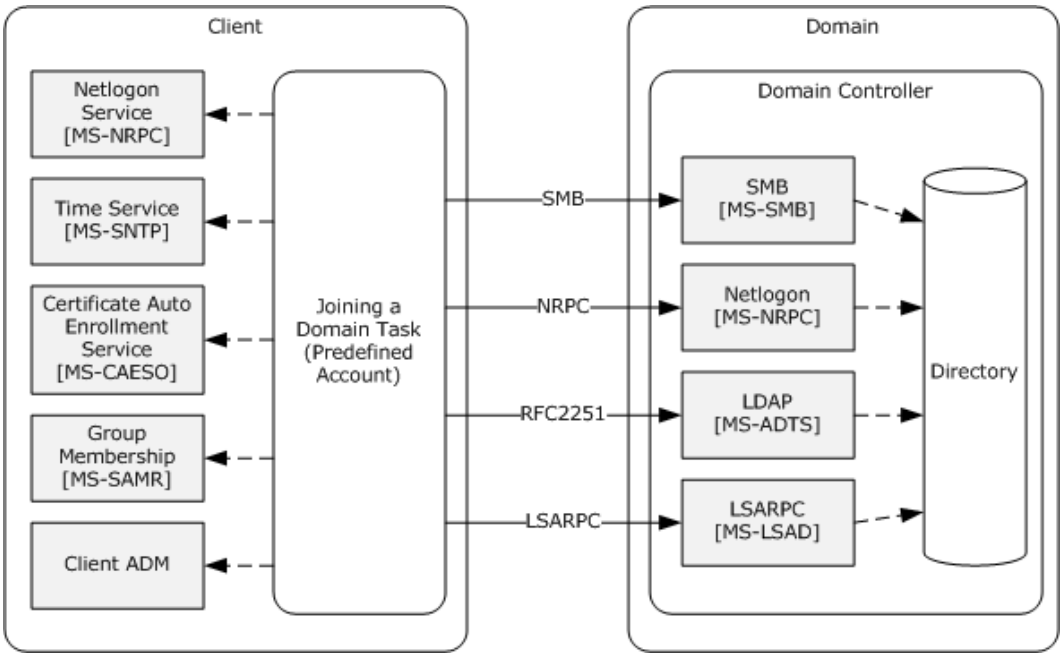

Figure 25: Task white-box relationships diagram

6.3.6 Task Events

6.3.6.1 Task Timers

None.

6.3.6.2 Task Non-Timer Events

None.

6.3.7 Task Architecture and Communication

Joining a Domain Task (Predefined Account): This box represents the task, including the task ADM and processing (see section 6.4.5). This box invokes the local [MS-NRPC] server to locate a domain controller. This box interacts with the domain controller for achieving the goals of this task. This box also configures client ADM, several client local states and fires required triggers as a final step in a successful run.

Client ADM, Group Membership: These boxes are configured by the Joining a Domain Task box as one of the final steps on a successful run of the task.

Time Service, Cert Auto Enrollment Service: These boxes are signaled by the Joining a Domain Task box as one of the final steps on a successful run of the task.

Netlogon Service: This box is configured by the Joining a Domain Task box as one of the final steps on a successful run of the task.

SMB, Netlogon, LDAP, LSARPC: These boxes represent the server-side implementations of the protocols used by this task, namely SMB, Netlogon, LDAP and LSAR.

Copyright © 2011 Microsoft Corporation.

Directory: This box represents the Directory Service running on the Domain Controller and it holds the predefined account for the client computer.

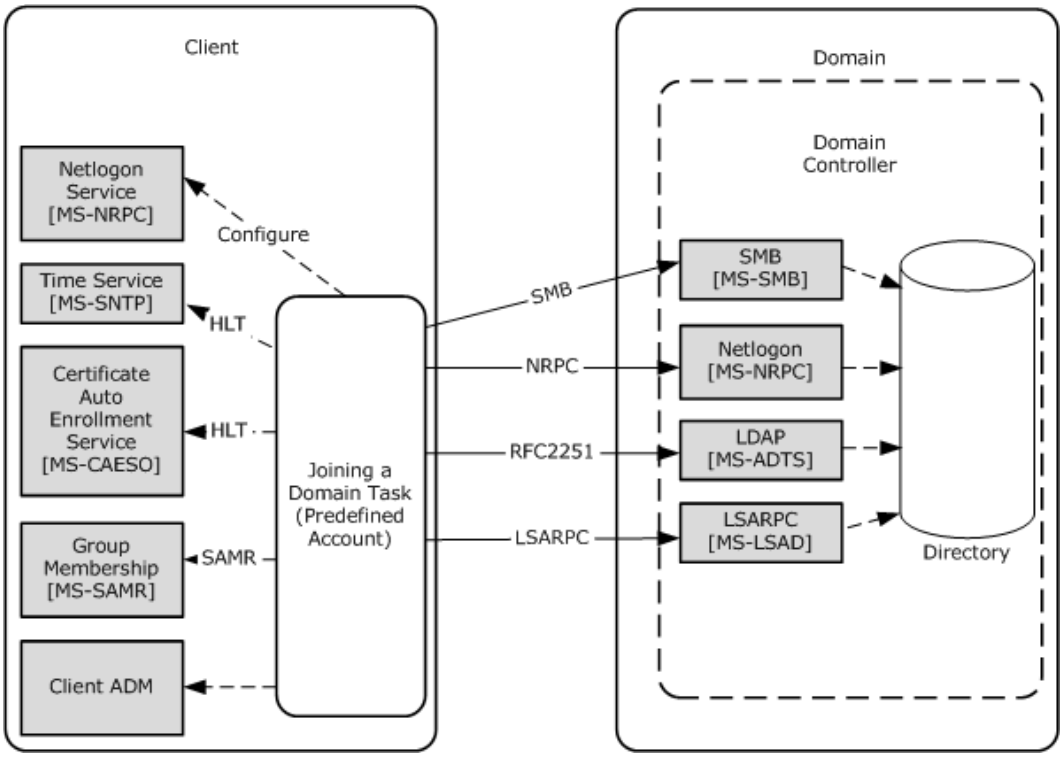

Figure 26: Client joining a domain using a predefined account – architecture and communication

6.3.8 Task Processing Rules

Abstract Parameters: As specified in section 6.3.3.

Preconditions:

- A predefined account is set up in the domain for joining the client to the domain.

Main Success Scenario:

1. Locate a domain controller to be used during task processing (see section 6.4.5.1).
2. Establish a SMB/CIFS session to the domain controller (see section 6.4.5.2).
3. Retrieve domain information from the domain controller using the [MS-LSAD] protocol (see section 6.4.5.3).
4. Validate the predefined account credentials using a Netlogon binding to the domain controller (see section 6.4.5.4).
5. Enumerate domain trusts (see section 6.4.5.5).
6. Update local state (see section 6.4.5.6).

Copyright © 2011 Microsoft Corporation.

7. Close the SMB/CIFS session (see section 6.4.5.7).

8. Reinitialize local protocols (see section 6.4.5.8).

Extensions:

1. Unable to locate a domain controller, terminate task with error.

2. Domain controller cannot be reached, terminate task with error.

3. Domain information cannot be retrieved, terminate task with error.

4. Netlogon binding fails, terminate task with error.

5. Domain trust enumeration fails, terminate task with error.

6. Update of local state fails, terminate task with error.

6.3.9 Task Failure Scenarios

The following are the common failure scenarios for the current task:

Unable to Locate DC: If a domain controller in the domain **TaskInputDomainName** cannot be located, (1), the task fails. This condition MAY occur if the network infrastructure is not configured correctly or is unavailable.

Domain Controller Unavailable: If **TaskLocalDomainController** cannot be reached due to any reason (2) while the client opens connections or queries or updates the domain controller (all the domain controller interactions shown), the task fails. This failure MAY happen because of conditions on the domain controller or network disruptions.

Authentication Fails: If the **TaskLocalClientName** and derived **TaskLocalPassword** do not authenticate the Client binding to the domain controller (4), the task fails. This failure happens if the predefined account has not been configured correctly.

Queries or Updates Fail on Domain Controller: If any of the queries or updates made to **TaskLocalDomainController** fails (3, 5), the task fails. This failure MAY happen because of conditions on the domain controller.

Failed Updates to Local State: If any of the updates made to local state fail (6), the task fails. This failure MAY happen because of conditions on the client.

6.4 Task Details

This section contains the details that complete the descriptions in earlier sections of the document. These are needed to understand and implement this task.

6.4.1 Task Precondition Details

Not applicable.

6.4.2 Task Initialization of External Entities

Once a domain controller is located using the Locate Domain Controller task, communication is directly with the domain controller.

Copyright © 2011 Microsoft Corporation.

6.4.3 Task Event Details

6.4.3.1 Task Timer Details

None.

6.4.3.2 Task Non-Timer Event Details

None.

6.4.4 Task Architectural Details

The general process flow is as follows: The client is operating in the environment described in section 6.2.1. The client tries to determine a domain controller in the domain it is required to join using **DsrGetDcNameEx2** method of the local MS-NRPC server. After the domain controller is determined, the client retrieves domain information using the LSARPC protocol. Then the client binds to the Netlogon RPC interface of the domain controller using the predefined account credentials, verifying its credentials in the process. The client then retrieves domain trusts from the domain controller. Finally, the client updates its own state saving data to indicate that it has now joined the domain.

The following figure shows the network traffic for a typical main success scenario. Note that the initial exchange (Locate a DC) is representative only of the traffic between the client and the selected domain controller; additional exchanges that may occur to other domain controllers are not represented. See section 5.4.4 for additional details.

The sequence diagram for this task is shown in the following figure.

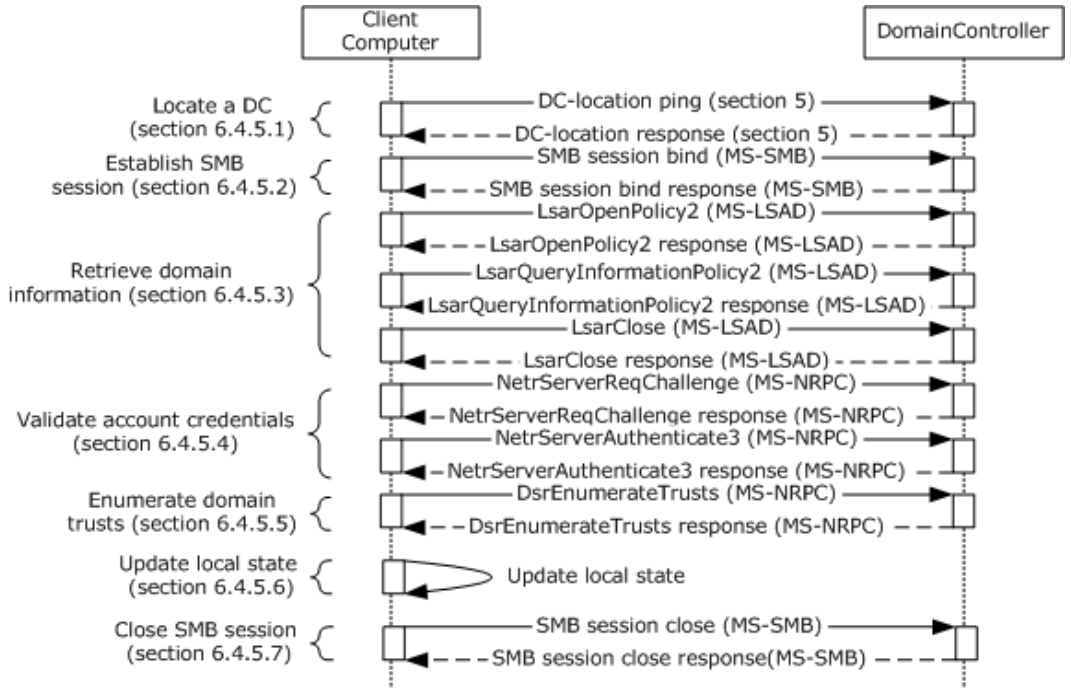

Figure 27: Joining a domain using a predefined account sequence diagram

Copyright © 2011 Microsoft Corporation.

6.4.5 Task Processing Rule Details

This section describes details for the steps identified in section 6.3.8. Unless otherwise specified, the processing falls through from one section to the next.

The following flowchart for the current task shows the sequential steps in the task, as well as failure decisions taken based on the results in each step.

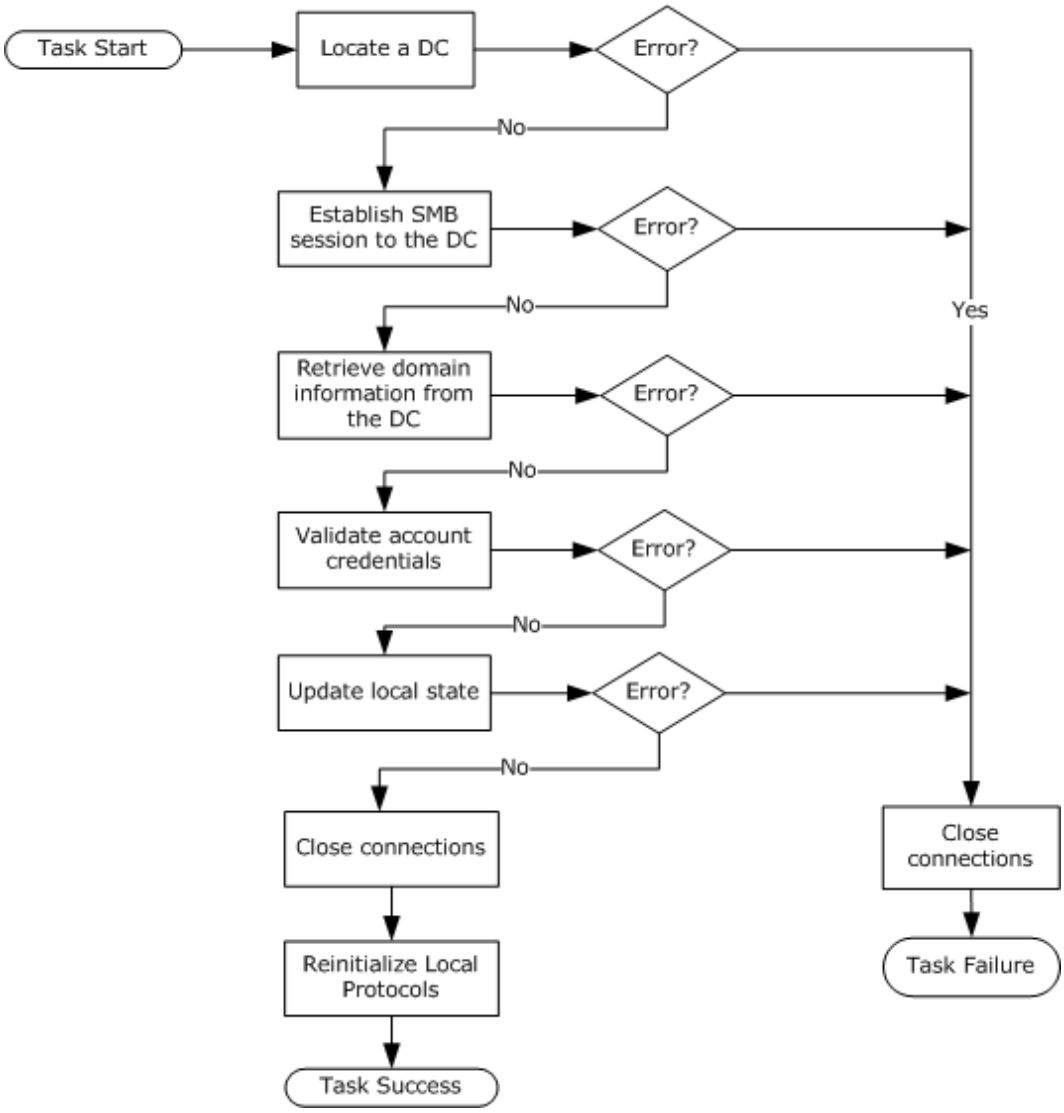

Figure 28: Joining a domain using a predefined account task flowchart

The initial state of the in-memory task ADM (see section 6.3.2) is initialized as follows:

```
TaskLocalClientName = ComputerName.NetBIOS "$"
TaskLocalPassword = ComputerName.NetBIOS in all lower case
TaskLocalNewPassword = <undefined>
TaskLocalDomainSID = <undefined>
TaskLocalDomainGUID = <undefined>
```


Copyright © 2011 Microsoft Corporation.

```
TaskLocalDomainController = TaskInputDomainController
TaskLocalClientAccountDN = <undefined>
TaskLocalDomainName = <undefined>
TaskLocalForestNameFQDN = <undefined>
TaskLocalSMBSession = <undefined>
TaskLocalAlreadyJoined = FALSE if DomainSID is equal to NULL, TRUE otherwise
```

6.4.5.1 Locate a Domain Controller

If **TaskLocalDomainController** is not already specified, the client MUST locate a domain controller as follows:

1. The client MUST invoke the **DsrGetDcNameEx2** method on the local MS-NRPC server, specifying the following parameters:

   - *ComputerName* = NULL
   - *AccountName* = **ComputerName.NetBIOS** (section 4.3.1.1)
   - *AllowableAccountControlBits* = ADS_UF_WORKSTATION_TRUST_ACCOUNT | ADS_UF_SERVER_TRUST_ACCOUNT ([MS-ADTS] section 2.2.15)
   - *DomainName* = **TaskInputDomainName**
   - *DomainGuid* = NULL
   - *SiteName* = NULL
   - *Flags* = (DS_WRITABLE_FLAG | DS_DS_FLAG | DS_LDAP_FLAG | DS_KDC_FLAG) ([MS-ADTS] section 7.3.1.2).

   Upon success, the following elements in the task ADM are set as follows:

   - **TaskLocalDomainController** = **DomainControllerInfo.DomainControllerName**

2. If the above **DsrGetDcNameEx2** call fails, the client MUST invoke the **DsrGetDcNameEx2** method on the local MS-NRPC server, specifying the following parameters:

   - *ComputerName* = NULL
   - *AccountName* = NULL
   - *AllowableAccountControlBits* = ADS_UF_WORKSTATION_TRUST_ACCOUNT | ADS_UF_SERVER_TRUST_ACCOUNT ([MS-ADTS] section 2.2.15)
   - *DomainName* = **TaskInputDomainName**
   - *DomainGuid* = NULL
   - *SiteName* = NULL
   - *Flags* = (DS_WRITABLE_FLAG | DS_DS_FLAG | DS_LDAP_FLAG | DS_KDC_FLAG) ([MS-ADTS] section 7.3.1.2).

   Upon success, the following elements in the task ADM are set as follows:

   - **TaskLocalDomainController** = **DomainControllerInfo.DomainControllerName**

Copyright © 2011 Microsoft Corporation.

If step 1 and step 2 both fail, the task MUST fail.

6.4.5.2 Establish SMB/CIFS Session to the Domain Controller

The client MUST establish an SMB/CIFS session using anonymous user credentials to the IPC$ share on the **TaskLocalDomainController** domain controller by invoking [MS-CIFS] section 3.4.4.7 specifying the following parameters:

ServerName = **TaskLocalDomainController.FQDN**

UserCredentials = Credentials of the anonymous user (consisting of an empty string "" for both username and password).

Upon success, the client MUST store the results in **TaskLocalSMBSession**.

6.4.5.3 Retrieve Domain Information from the Domain Controller

The SMB/CIFS session established to **TaskLocalDomainController** in the preceding section is reused as transport for LSA RPC calls as per the algorithm defined in [MS-CIFS] section 3.2.4.2.1.

1. The client MUST bind to the LSA RPC endpoint, as specified in [MS-LSAD] section 1.9, on **TaskLocalDomainController.FQDN** with the same credentials used to establish the SMB session in the previous step.

   The bind can fail when using anonymous user credentials and the domain controller is not configured to accept anonymous clients using the LSA RPC endpoint (see [MS-SRVS] sections 3.1.3 and 3.1.6.17). The client MUST use a more secure algorithm (section 7 or section 8) to join the domain in that case.

2. The client MUST invoke the **LsarOpenPolicy2** method ([MS-LSAD] section 3.1.4.4.1) with the following parameter values:

   SystemName = **TaskLocalDomainController.FQDN**

   ObjectAttributes = NULL

   DesiredAccess = MAXIMUM_ALLOWED

   Upon success, **LsarOpenPolicy2** returns as an out parameter a *PolicyHandle* (context handle), which MUST be used for the subsequent **LsarQueryInformationPolicy2** and **LsarClose** calls.

3. The client MUST invoke **LsarQueryInformationPolicy2** method ([MS-LSAD] section 3.1.4.4.3) with the following parameter values:

   InformationClass = PolicyDnsDomainInformation

4. The client MUST update the task ADM (section 6.3.2) as follows:

   TaskLocalDomainName.NetBIOS = PolicyInformation.PolicyDnsDomainInfo.Name

   TaskLocalDomainName.FQDN = PolicyInformation.PolicyDnsDomainInfo.DnsDomainName

   TaskLocalDomainSID = PolicyInformation.PolicyDnsDomainInfo.Sid

   TaskLocalDomainGUID = PolicyInformation.PolicyDnsDomainInfo.Guid

   TaskLocalForestNameFQDN = PolicyInformation.PolicyDnsDomainInfo.DnsForestName

Copyright © 2011 Microsoft Corporation.

The client MUST invoke **LsarClose** ([MS-LSAD] section 3.1.4.9.4) to close the handle opened in step 2.

5. Any failure during the preceding call sequence is will cause the task to fail.

6.4.5.4 Validate the Predefined Account Credentials

The next step is validating the predefined account credentials by invoking the **NetrServerReqChallenge** and **NetrServerAuthenticate3** [MS-NRPC] methods on **TaskLocalDomainController**. The existing SMB/CIFS session MUST be used as the Netlogon transport.

- The client MUST invoke the **NetrServerReqChallenge** method on the domain controller, specifying the following parameters:

  PrimaryName = **TaskLocalDomainController**

  ComputerName = **ComputerName.NetBIOS**

  ClientChallenge = (set as specified in [MS-NRPC] section 2.2.1.3.4 and 3.4.5.2.1)

- The client MUST invoke the **NetrServerAuthenticate3** method on the domain controller, specifying the following parameters:

  PrimaryName = **TaskLocalDomainController**

  AccountName = **ComputerName.NetBIOS**$

  SecureChannelType = WorkstationSecureChannel (see [MS-NRPC] section 2.2.1.3.12)

  ComputerName = **ComputerName.NetBIOS**

  ClientCredential = (set as specified in [MS-NRPC] section 2.2.1.3.4)

  NegotiateFlags = O bit (as specified in [MS-NRPC] section 3.1.4.2)

When the authentication sequence finishes successfully, the client and domain controller are assured of each other's identity and the credentials of the predefined account are validated as well. The predefined account can henceforth be referred to as "client account" or "client computer account" and the three terms can be used interchangeably to refer to the same account in the directory.

Any failure during the preceding sequence MUST cause the task to fail.

6.4.5.5 Enumerate Domain Trusts

The client MUST retrieve the list of Domain Trusts from the domain controller by invoking the **DsrEnumerateDomainTrusts** method ([MS-NRPC] section 3.5.5.6.1)<10> specifying the following parameters:

ServerName = **TaskLocalDomainName.FQDN**

Flags = A|B|C|D|E|F ([MS-NRPC] section 3.5.5.6.1)

Upon successful completion of this stage, the client MUST update the following ADM elements:

TaskLocalTrustedDomains = List of domain trusts returned by **DsrEnumerateDomainTrusts**

Copyright © 2011 Microsoft Corporation.

6.4.5.6 Update Local State

1. Update local state.

    1.a. The client MUST stop the Netlogon Remote Protocol (MS-NRPC) if it is running.

    If this step fails, the task fails. Otherwise execution continues at step 1.b.

    1.b. If **TaskLocalAlreadyJoined** is TRUE, the client MUST invoke the "Certificate Autoenrollment Task" task ([MS-CAESO] section 4) specifying the *IsUnjoiningDomain* input parameter ([MS-CAESO] section 4.3.3) to be TRUE.

    The task must continue execution at the next step (1.c) regardless if this step failed.

    1.c. If **TaskLocalAlreadyJoined** is TRUE, the client MUST invoke the domain unjoin processing event in [MS-SAMR] section 3.1.7.2 with the parameters set as follows:

    DomainSID = **DomainSID** (section 4.3.1.1)

    The task must continue execution at the next step (1.d) regardless if this step failed.

    1.d. The client MUST invoke the domain join processing event in [MS-SAMR] section 3.1.7.1 with the parameters set as follows:

    DomainSID = **TaskLocalDomainSID**

    If this step fails, the task must begin executing step 2 as follows.

    1.e. The client MUST update the following client state values to the ADM values specified in this section. The client MUST remember the original values in case they need to be restored in case of a failure (see step 2.a). These values MUST be modified in a single atomic update.

    DomainName.FQDN = **TaskLocalDomainName.FQDN**

    DomainName.NetBIOS = **TaskLocalDomainName.NetBIOS**

    DomainGuid = **TaskLocalDomainGUID**

    DomainSid = **TaskLocalDomainSID**

    ForestNameFQDN = **TaskLocalForestNameFQDN**

    ClientName = **TaskLocalClientName**

    Password = **TaskLocalPassword**

    TrustedDomains = **TaskLocalTrustedDomains**

    If this step fails, the task must begin executing step 2 below.

    1.f. The client MUST configure the Netlogon Remote Protocol to start automatically at every boot.

    If this step fails, the task must begin executing step 2 below.

    1.g. The client MUST start the NetLogon Remote protocol.

    If this step fails, the task must begin executing step 2 below.

Copyright © 2011 Microsoft Corporation.

1.h. The client MUST invoke the Domain Join Processing higher layer triggered event in [MS-SNTP] section 3.1.4.1; the result of this operation is ignored by the task.

2. Rollback steps in case of error.

The following steps are executed only upon a failure of one of the substeps shown in section 1.

2.a. The client MUST restore the client ADM variables updated in step 1.c above to their original values.

2.b. If **TaskLocalAlreadyJoined** is TRUE, the client MUST configure the Netlogon Remote Protocol to start automatically at every boot; otherwise, the client MUST configure the Netlogon Remote Protocol to not start automatically at every boot.

2.c. If **TaskLocalAlreadyJoined** is TRUE , the client MUST initialize the Netlogon Remote Protocol; otherwise, the client MUST uninitialize the Netlogon Remote Protocol.

2.d. If **TaskLocalAlreadyJoined** is TRUE, and if step 1.c succeeded, the client MUST invoke the Domain join processing event in MS-SAMR ([MS-SAMR] section 3.1.7.1) specifying the following parameters:

DomainSID = **DomainSID** (section 4.3.1.1)

2.e. If step 1d succeeded, the client MUST invoke the Domain unjoin processing event in MS-SAMR ([MS-SAMR] section 3.1.7.2) specifying the following parameters:

DomainSID = **TaskLocalDomainSID**

2.f. If step 1b succeeded, the client MUST invoke the "Certificate Autoenrollment Task" task ([MS-CAESO] section 4) specifying the IsUnjoiningDomain input parameter ([MS-CAESO] section 4.3.3) to be FALSE.

6.4.5.7 Close Connections

If an SMB/CIFS session was previously established (see section 6.4.5.4), the client MUST disconnect as described in [MS-CIFS] section 3.4.4.8 specifying **TaskLocalSMBSession**.

6.4.5.8 Reinitialize Local Protocols

If the task succeeded, the client MUST reinitialize the [MS-SNTP] protocol. The client MAY do this by rebooting itself, or by any other supported mechanism.<11>

6.5 Task Security

This task is *not secure* by definition, as the password is derived from the **ComputerName.NetBIOS** and not exchanged in a secure manner. Sections 7 and 8 describe more secure versions of domain join.

Upon successful completion of this task by a client already joined to a domain, the client's machine account in the old domain will be left as it is. The administrator of the old domain should consider disabling or deleting the old machine account as a security best practice.

Please refer to the Security section of this specification and the Security sections of the referenced protocol Technical Documents (TDs) for protocol-specific security issues.

Copyright © 2011 Microsoft Corporation.

7 Joining a Domain by Creating an Account via SAMR

This section describes the process of joining a client computer to a domain by creating an account via the MS-SAMR protocol. This task shares many of the other actions in establishing the relationship between the client and the domain controller with the task in section 6, which will be cross referenced where appropriate.

7.1 Task Overview

7.1.1 Task Purpose

The purpose of this task is to securely join a client computer to a domain by creating an account for the client computer as part of the domain join via [MS-SAMR].

7.1.2 Task Applicability

This task is applicable to a client attempting to join a domain securely using the MS-SAMR protocol. The credentials of a Domain Administrator are required to perform this task.

7.1.3 Task Use Cases

7.1.3.1 Stakeholders and Interests Summary

Client Administrator: The client administrator is the administrator of the client computer, interested in joining the client computer to the domain.

Client Computer: The client computer that is being joined to the Domain.

Domain Administrator: The domain administrator is the administrator of the Domain. The Domain Administrator supplies the credentials to this task for creating and configuring the machine account on the domain controller. After the join is completed, the Domain Administrator is able to influence the client computer via other protocols; see sections 3.1 and 3.4, and [MS-WSO], for further details.

Domain Controller: The domain controller is a computer providing domain services to domain clients. The client creates and configures a machine account on the domain controller during task processing.

7.1.3.2 Supporting Actors and Task Interests Summary

This Task depends on the following supporting actors for the specified interests:

- SMB protocol ([MS-SMB2], [MS-SMB], or [MS-CIFS]), for opening/closing SMB sessions to a DC. The Task does not require that any specific SMB protocol be used.
- [MS-SAMR] protocol, for creating a client computer account on a DC.
- [MS-NRPC] protocol, for locating a DC, for establishing a Netlogon binding to a DC, and for enumerating the trusted domains, and for updating the machine account password.
- [MS-LSAD] protocol, for retrieval of information about a domain from a DC.
- [MS-DRDM] protocol, for searching for the client computer account on a DC.
- [RFC2251] protocol, for updating attributes on the client computer account on a DC.

Copyright © 2011 Microsoft Corporation.

There are no other systems or tasks in which this Task is an actor.

7.1.3.3 Use Case Diagrams

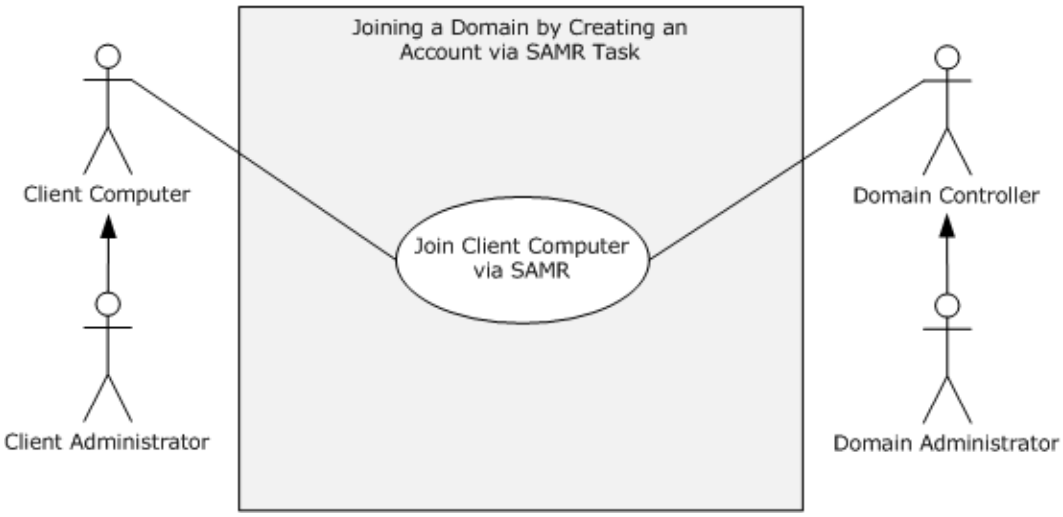

Figure 29: Use case diagram; join a domain by creating an account via SAMR

7.1.3.4 Join a Client Computer to a Domain by Creating an Account via SAMR — Client Computer

The only use case for this task is a client **administrator** joining the client computer to a domain by providing **domain administrator's credentials**. The client creates an account for the client computer in the domain via [MS-SAMR] using domain administrator's credentials.

Goal: Join a client computer to a domain by creating an account for the client computer in the domain via MS-SAMR.

Context of Use: This task is invoked by the client administrator in order to enable the client computer to access the services and resources in a domain, as well as provide the domain members access to the client computer. See sections 3.1 and 3.4 for other motivations for joining a domain.

Primary Actor: The client administrator is the primary actor who initiates joining the client computer to a domain. The client administrator invokes commands/applications on the client that initiate the current task.

Direct Actor: The client computer joining the domain.

Supporting Actors: All supporting actors are specified in section 7.1.3.2.

Stakeholders and Interests: There are no other stakeholders for the current task besides those listed under Primary Actor, Direct Actor, and Supporting Actors sections as presented earlier.

Precondition: The preconditions for this use case are the same as those listed for the task in section 7.2.3.

Minimal Guarantees: this use case guarantees that upon failure, the local client computer state defined in section 4.3.1.1 is unchanged.

Success Guarantee: The client is joined to the domain.

Copyright © 2011 Microsoft Corporation.

Trigger: This use case is triggered by the client administrator to join the client to a domain.

Main Success Scenario:

1. The client administrator initiates the domain join task on the client computer.
2. The client locates a domain controller using the Locate a Domain Controller Task (see section 5).
3. The client opens a secure connection to the domain controller using Domain Administrator's supplied credentials and retrieves domain information.
4. The client uses domain administrator's credentials and sets up an account in the domain for itself.
5. The client determines the trusted domains.
6. The client updates the client account in the domain.
7. The client updates the local client state.
8. The client reinitializes local protocols.

Extensions: None.

7.2 Task Context

This section describes the relationship between this Task and its environment.

7.2.1 Task Environment

The task environment is the same as described in section 6.2.1.

7.2.2 Task Relationships

This task builds on the Common Task Information (see section 4) which is shared with all of the tasks in this document.

7.2.2.1 Black Box Relationship Diagrams

The following diagram illustrates the system as the client computer joins the domain by creating a new account on the domain controller via the Security Account Manager RPC ([MS-SAMR]).

Copyright © 2011 Microsoft Corporation.

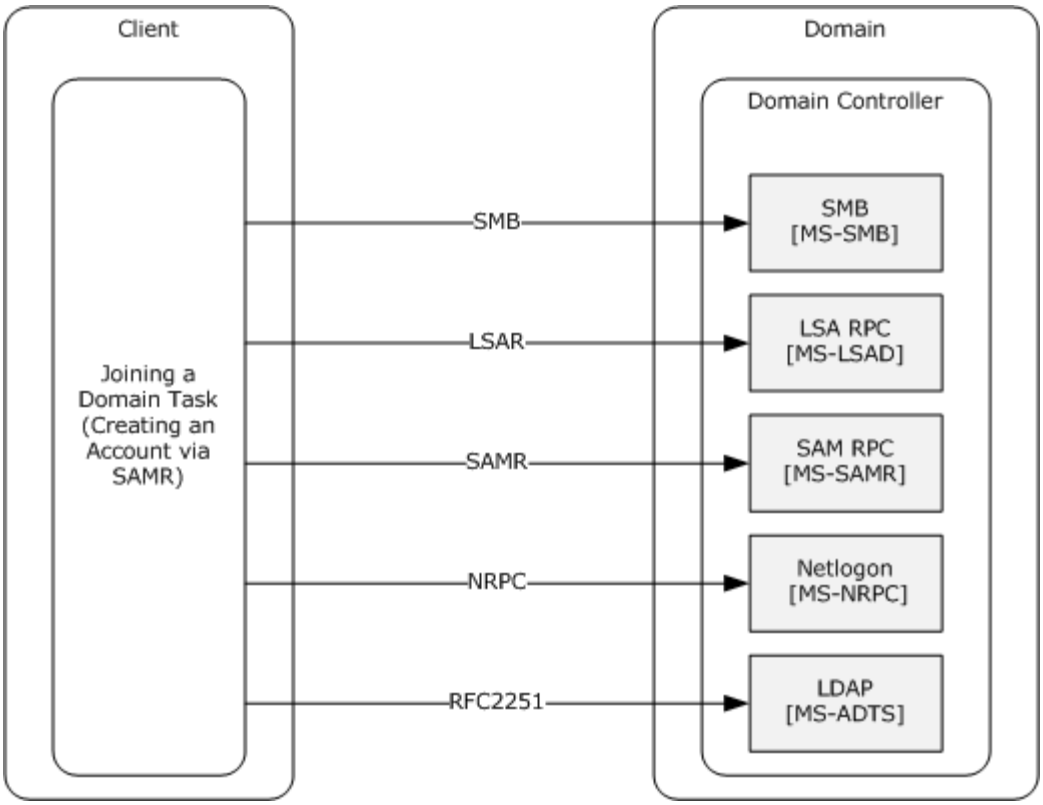

Figure 30: Client machine joining a domain by creating an account via SAM

7.2.2.2 Task Dependencies

This task depends on the following:

- At least one domain controller that is available.
- A domain controller that is located using the **DsrGetDcNameEx2** method of the local [MS-NRPC] server.

7.2.2.3 Task Influences

None.

7.2.3 Task Assumptions and Preconditions

The following are the success conditions for this task:

- The credentials of an Administrator of the Domain who can create machine accounts in the domain are available to the Client Administrator.

7.2.4 Task Versioning and Capability Negotiation

None.

Copyright © 2011 Microsoft Corporation.

7.3 Task Architecture

7.3.1 Task Architectural Constraints

This task has the following architectural constraints:

- The client computer can be joined to only a single domain at a time.
- Only one instance of this task MUST run on the client computer at any given time. As the task aims to join the client computer to a single domain, it does not support multiple, parallel executions.
- This task MAY be run a second time following a previous successful completion, in order to join the client computer to a different domain.
- This task MAY be run a second time following a previous unsuccessful run, as an unsuccessful run makes no changes to the client state.
- This task MUST NOT make assumptions about distributed state, such as the machine account on the domain controller.

7.3.2 Task Abstract Data Model

This section describes state established, used, and maintained by processing rules of this Task. State may be volatile or persisted. State may pertain to one, some, or all instances of the Task. The Task's state consists of the values of the named data elements (also called state variables) presented in this section. The overall organization of the data elements, with their names, is the Abstract Data Model. It is intended to facilitate the reader's conceptual understanding of the specification. While a Task's processing rules may depend upon associations established by the structure of its Abstract Data Model, such association can be achieved in other ways. Implementations may depart from this model so long as their external behavior remains consistent with that described in this document.

The following are the in-memory values used by the task. These values are not persisted.

Name	Type	Description
TaskLocalClientName	string (Unicode)	Contains the **samAccountName** attribute value for the account in the domain.
TaskLocalPassword	string (Unicode)	Contains the password for the account in the domain.
TaskLocalNewPassword	string (Unicode)	Contains a randomly generated password.
TaskLocalDomainName.FQDN	string (Unicode)	Contains the FQDN(1) of the domain being joined.
TaskLocalDomainName.NetBIOS	string (Unicode)	Contains the NetBIOS name of the domain being joined.
TaskLocalForestNameFQDN	string (Unicode)	Contains the FQDN(1) of the forest containing the domain being joined.
TaskLocalDomainController	string (Unicode)	Contains the name of the domain controller used by this task.

Copyright © 2011 Microsoft Corporation.

Name	Type	Description
TaskLocalDomainSID	SID	Contains the SID of the domain being joined.
TaskLocalDomainGUID	GUID	Contains the guid of the domain being joined.
TaskLocalTrustedDomains	Domains	Contains a list of trusted domains for the domain being joined.
TaskLocalClientAccountDN	string (Unicode)	Contains the Distinguished Name of the client account in the domain.
TaskLocalSMBSession	SMB/CIFS session	Contains the returned SMB state for an SMB/CIFS session.
TaskLocalAlreadyJoined	Boolean	Whether the client is already joined to a domain at the start of the task.

The following client data model variables (section 4.3.1.1) are updated by the task during a successful completion:

- **DomainSid**
- **DomainGuid**
- **DomainName** (NetBIOS and FQDN Names)
- **ClientName**
- **Password**
- **TrustedDomains**

7.3.3 Task Abstract Parameters

This section describes data passed to an instance of this task at the time it is invoked or triggered. The parameters consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Parameter. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules could depend upon associations established by the structure of its Abstract Parameters, such association can be achieved in other ways. Implementations can depart from this abstraction so long as their external behavior remains consistent with that described in this document.

The parameters to this task are as follows:

Name	Type	Description	Optional
TaskInputDomainName	string (Unicode)	Specifies the NetBIOS or FQDN name of the domain to join.	No
TaskInputDomainController	string (Unicode)	Specifies the domain controller this task must use to complete the join.	Yes
TaskInputDomainAdminAccount	string (Unicode)	Specifies the name of a Domain Administrator account.	No
TaskInputDomainAdminPassword	string	Specifies the password for the	No

Copyright © 2011 Microsoft Corporation.

Name	Type	Description	Optional
	(Unicode)	TaskInputDomainAdminAccount.	

7.3.4 Task Abstract Results

This section describes data returned by an instance of this task to its caller. The results consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Result. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules might depend upon associations established by the structure of its Abstract Results, such association can be achieved in other ways. Implementations can depart from this abstraction so long as their external behavior remains consistent with that described in this document.

The task returns the following results to the caller:

Name	Type	Description
TaskReturnStatus	integer	This task MUST return 0x00000000 on success. Error statuses generated by a failure during task processing are in the Win32 error space (a long data type), as specified in [MS-ERREF] section 2.2.

Upon a successful task completion this task performs the following steps:

1. Updates client ADM variables (section 4.3.1.1). The updated values are persisted, as specified in section 4.3.1.1, and are available for other protocols on the client to use.
2. Starts the **NetLogon** Remote Protocol and configures it to run automatically at every system start up.
3. Notifies the Certificate Autoenrollment system that the machine is joining the domain.
4. Adds the Domain Administrators to the local Administrators group.
5. Notifies the local MS-SNTP protocol that the machine is joining the domain (failures in this step are ignored).
6. Creates or modifies a client computer account in the domain.

Upon a failed task completion, this task will attempt to disable the computer account in the domain (only if the task created the account, as opposed to modifying an existing account), and will attempt to revert any local state changes as well. If the task is unable to disable the computer account upon failure, domain administrator intervention outside the scope of the task may be necessary to disable the account. If the task is unable to revert local state changes, administrator intervention outside the scope of the task may be necessary to reconfigure the client back to its original state.

7.3.5 White-Box Relationships

The following diagram represents the white-box relationships of the task within the client computer and with the domain controller.

Copyright © 2011 Microsoft Corporation.

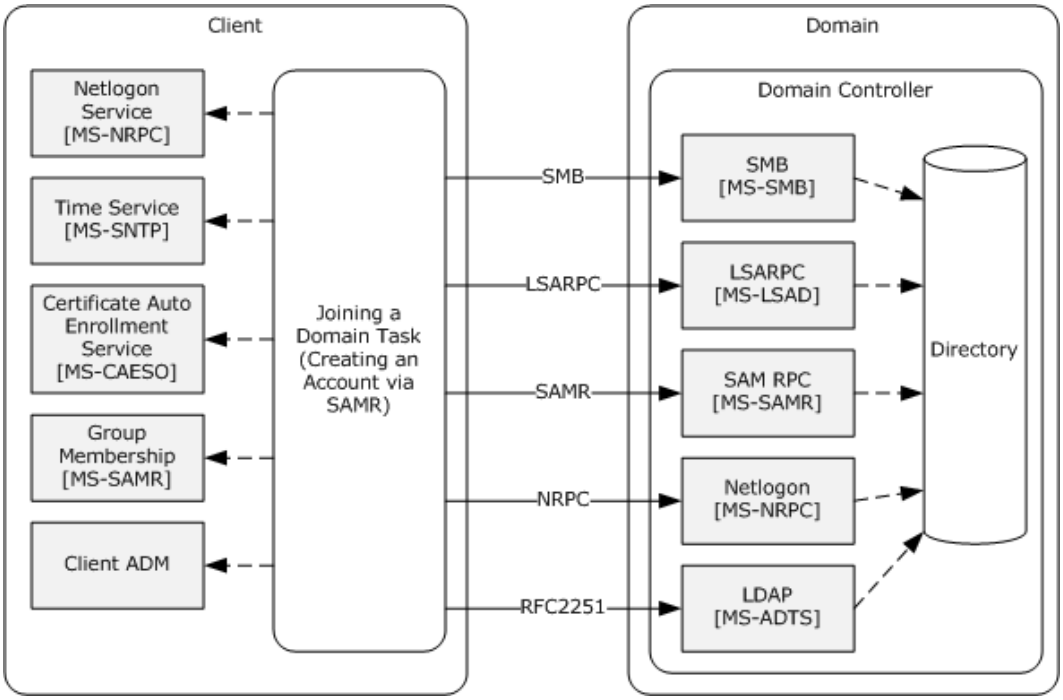

Figure 31: Task white-box relationships diagram

7.3.6 Task Events

7.3.6.1 Task Timers

None.

7.3.6.2 Task Non-Timer Events

None.

7.3.7 Task Architecture and Communication

The following paragraphs document the boxes in the following diagram.

Joining a Domain Task (Creating an Account via SAMR): This box represents the task, including the task ADM and processing (see section 7.4.5). This box invokes the local [MS-NRPC] server to locate a domain controller. This box interacts with the domain controller for achieving the goals of this task. This box also configures client ADM, several client local states and fires required triggers as a final step in a successful run.

Client ADM, Group Membership: These boxes are configured by the Joining a Domain Task box as one of the final steps on a successful run of the task.

Time Service, Cert Auto Enrollment Service: These boxes are signaled by the Joining a Domain Task box as one of the final steps on a successful run of the task.

Netlogon Service: This box is configured by the Joining a Domain Task box as one of the final steps on a successful run of the task.

Copyright © 2011 Microsoft Corporation.

SMB, Netlogon, LDAP, LSARPC, SAM RPC: These boxes represent the server-side implementations of the protocols used by this task, namely SMB, Netlogon, LDAP, LSAR and SAM RPC.

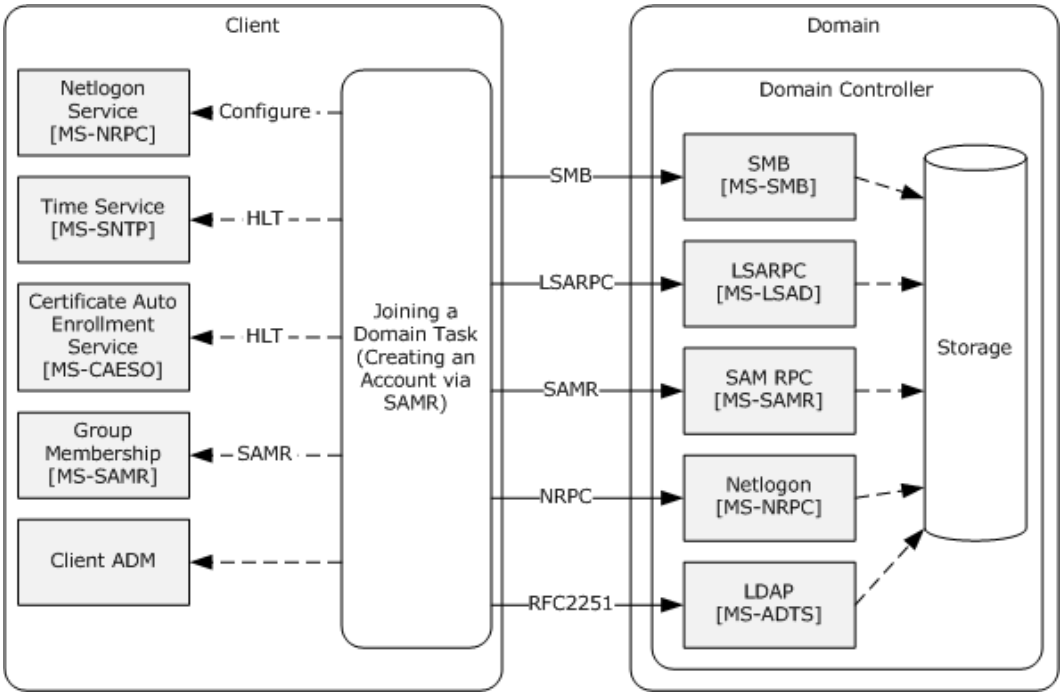

Figure 32: Client joining a domain (creating an account via SAMR) – architecture and communication

7.3.8 Task Processing Rules

Abstract Parameters: As specified in section 7.3.3.

Main Success Scenario:

1. Locate a domain controller to be used during task processing (see section 7.4.5.1).
2. Establish an SMB/CIFS session to the domain controller (see section 7.4.5.2).
3. Retrieve domain information from the domain controller (see section 7.4.5.3).
4. Create client computer account on the domain controller (see section 7.4.5.4).
5. Enumerate domain trusts (see section 7.4.5.6).
6. Update client computer account on the domain controller (see section 7.4.5.5).
7. Update local state (see section 7.4.5.7).
8. Close SMB/CIFS session (see section 7.4.5.9).
9. Reinitialize local protocols (see section 7.4.5.10).

Extensions:

Copyright © 2011 Microsoft Corporation.

1. Unable to locate a domain controller, terminate task with error.
2. Domain controller cannot be reached, terminate task with error.
3. Domain information cannot be retrieved, terminate task with error.
4. Computer account cannot be created, terminate task with error.
5. Domain trust enumeration fails, attempt to disable the computer account on the DC and terminate task with error.
6. Account update fails, attempt to disable the computer account on the DC and terminate task with error.
7. Local state updates fail; attempt to disable the computer account on the DC and terminate task with error.

7.3.9 Task Failure Scenarios

The following are the common failure scenarios for the current task:

Unable to Locate DC: If a domain controller in the domain **TaskInputDomainName** cannot be located, (1). The result is that the join task fails. This condition MAY occur if the network infrastructure is not configured correctly or is unavailable.

Domain Controller Unavailable: If **TaskLocalDomainController** cannot be reached due to any reason while the client opens connections or queries or updates the domain controller (all the domain controller interactions), the join task fails. This failure MAY happen because of conditions on the domain controller or network disruptions.

Authentication Fails: If the **TaskInputDomainAdminAccount** and derived **TaskInputDomainAdminPassword** do not authenticate the Client binding to the server (3, 4), further processing is not possible and the join task fails.

Queries or Updates Fail on Domain Controller: If any of the queries or updates made to **TaskLocalDomainController** fails (2, 3, 4, 5, 6), the join task fails. This failure MAY happen because of conditions on the domain controller.

Update fails on Client: Update to client local state fails (7) and as a result the join task fails. This failure MAY happen because of conditions on the client computer.

7.4 Task Details

This section contains the details that complete the descriptions in earlier sections of the document. These are needed to understand and implement this Task.

7.4.1 Task Precondition Details

Not applicable.

7.4.2 Task Initialization of External Entities

Once a domain controller is located using the **DsrGetDcNameEx2** method of the local [MS-NRPC] server, communication is directly with the domain controller.

Copyright © 2011 Microsoft Corporation.

7.4.3 Task Event Details

7.4.3.1 Task Timer Details

None.

7.4.3.2 Task Non-Timer Event Details

None.

7.4.4 Task Architectural Details

This task is a secure variant of that described in section 6 for joining a domain. The general process flow is as follows: The client is operating in the environment described in section 7.2.1. The client attempts to determine a domain controller in the domain it is required to join using the **DsrGetDcNameEx2** method of the local [MS-NRPC] server. When the domain controller is determined, the client binds to the domain controller through a secure SMB authenticated using the domain administrator's credentials and retrieves various pieces of domain information. The client creates an account on the domain controller for itself using the domain administrator's credentials using the [MS-SAMR] protocol. The client then retrieves domain trusts from the domain controller. The client then updates attributes on the client computer account on the domain controller. Finally, the client updates its own state to indicate that it has now joined the domain. The following figure shows the sequence diagram for this task, except for the shared steps, which can be referred to in section 6.4.4. For details about locating a domain controller, refer to [MS-NRPC].

The following figure shows the network traffic for a typical main success scenario. Note that the initial exchange (Locate a DC) is representative only of the traffic between the client and the selected domain controller; additional exchanges that might occur to other domain controllers are not represented. See [MS-NRPC] for additional details.

Copyright © 2011 Microsoft Corporation.

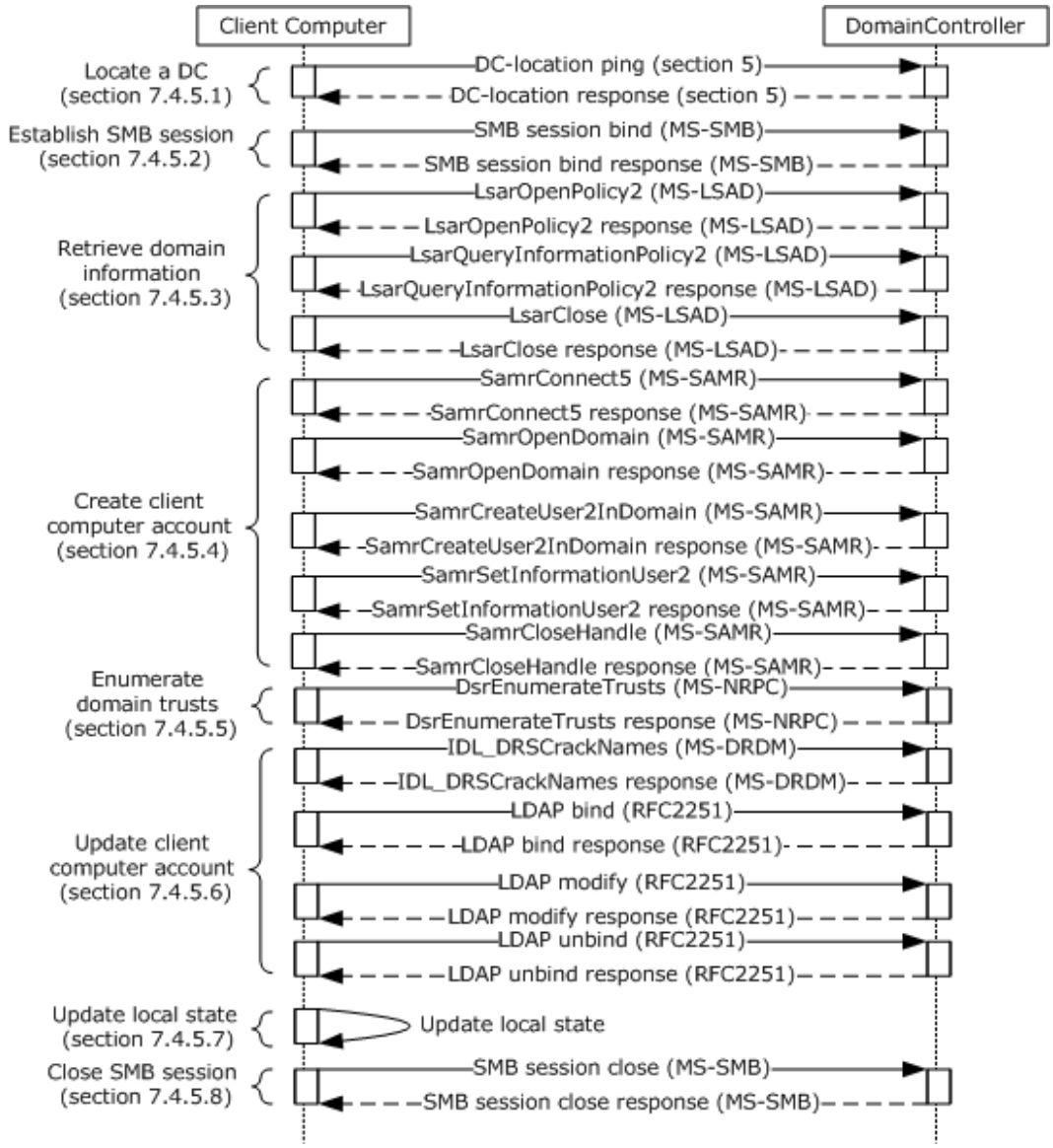

Figure 33: Joining a domain (creating account via SAM) task architectural details

7.4.5 Task Processing Rule Details

This section describes details for the steps identified in section 7.3.8. Unless otherwise specified, the processing falls through from one section to the next.

The following flowchart for the current task shows the sequential steps in the task, as well as failure decisions taken based on the results in each step.

Copyright © 2011 Microsoft Corporation.

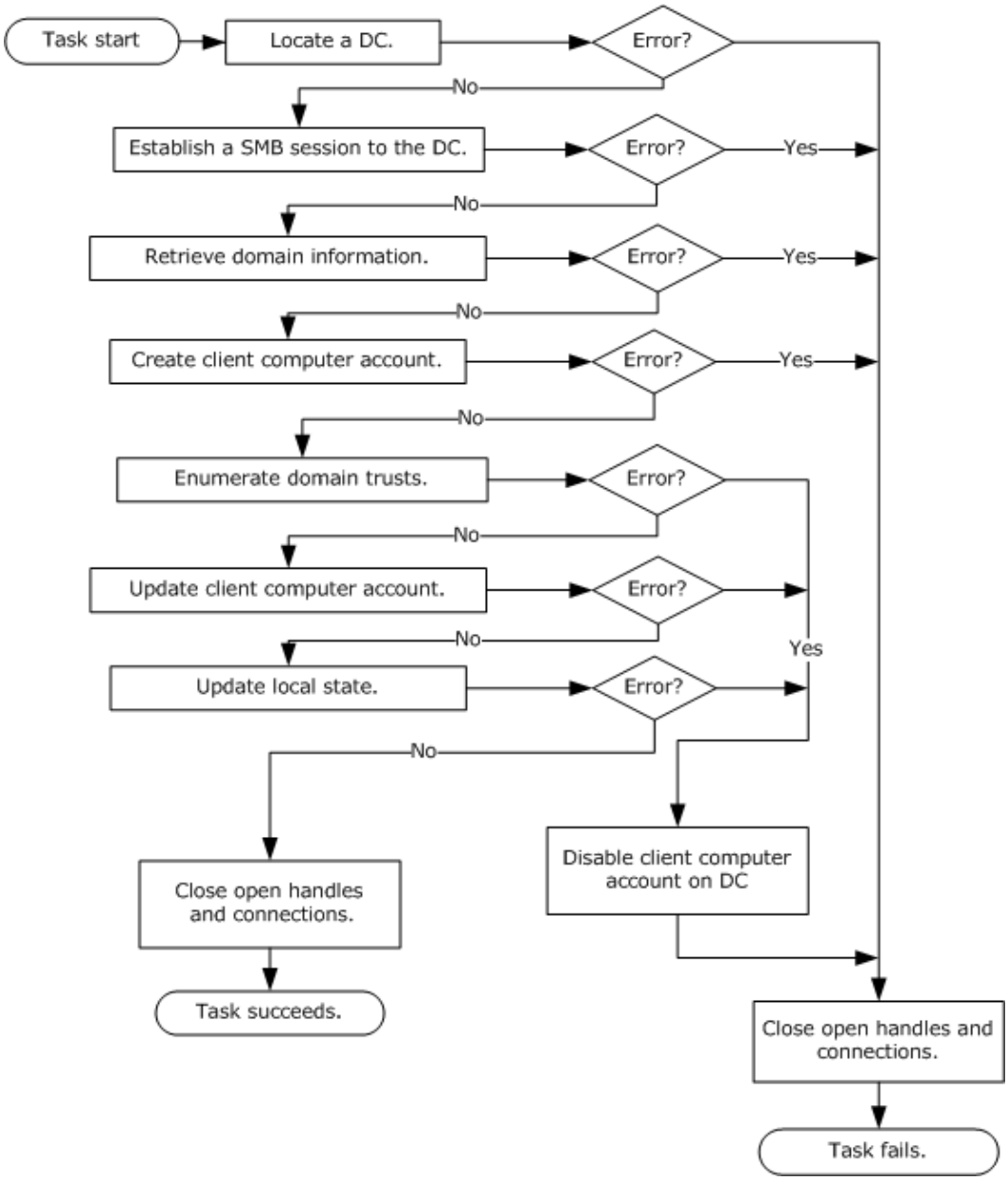

Figure 34: Joining a domain (creating account via SAM) task flowchart

The abstract parameters specified in section 7.3.3 are passed to this task upon entry.

The initial state of the in-memory task ADM is initialized as follows:

```
TaskLocalClientName = ComputerName.NetBIOS "$"
TaskLocalPassword = <undefined>
TaskLocalNewPassword = <undefined>
TaskLocalDomainSID = <undefined>
TaskLocalDomainGUID = <undefined>
TaskLocalDomainController = TaskInputDomainController
```


Copyright © 2011 Microsoft Corporation.

```
TaskLocalClientAccountDN  = <undefined>
TaskLocalDomainName = <undefined>
TaskLocalForestNameFQDN = <undefined>
TaskLocalSMBSession = <undefined>
TaskLocalAlreadyJoined = FALSE if DomainSid is equal to NULL, TRUE otherwise
```

7.4.5.1 Locate a Domain Controller

This step of the task processing is the same as section 6.4.5.1, with the same actions on success and failure outcomes as described in that section.

7.4.5.2 Establish Authenticated SMB Session

The client MUST establish an authenticated SMB/CIFS session to the IPC$ share on the **TaskLocalDomainController** domain controller by invoking [MS-CIFS] section 3.4.4.7 specifying the following parameters:

ServerName = **TaskLocalDomainController.FQDN**

UserCredentials = **TaskInputDomainAdminAccount** \ **TaskInputDomainAdminPassword**

Upon success, the client MUST store the results in **TaskLocalSMBSession**.

7.4.5.3 Retrieve Domain Information

The client MUST retrieve domain information from the domain controller, over the secure SMB/CIFS session established in the preceding step, by using the steps described in section 6.4.5.3.

7.4.5.4 Create Client Computer Account

The client MUST use the authenticated SMB connection established in the prior step as a transport for remote calls to the domain controller in this section. The client MUST create an account for the client computer in the domain using the following steps:

1. The client MUST bind to the named pipe endpoint \PIPE\samr, as shown in [MS-SAMR] section 2.1.

2. The client MUST connect to the SAM RPC server on the domain controller using one of the **SamrConnect** variants. See [MS-SAMR] section 1.7.2 for information about invoking the **SamrConnect** variants in order to the determine version and method supported by the server. See [MS-SAMR] section 3.1.5.1 for using the Open pattern in the SAM interface.

   ServerName = **TaskLocalDomainController.FQDN**

   DesiredAccess = GENERIC_ALL

3. The client MUST call SamrOpenDomain ([MS-SAMR] section 3.1.5.1.5) specifying the following parameters:

   DesiredAccess = GENERIC_ALL

   DomainSid = **TaskLocalDomainSID** from the task ADM.

4. The client MUST attempt to create the account on the domain controller. If this fails because of an existing account, the client MUST attempt to obtain a handle to the existing account.

Copyright © 2011 Microsoft Corporation.

The client MUST call **SamrCreateUser2InDomain** ([MS-SAMR] section 3.1.5.4.4) specifying the following parameters:

Name = **TaskLocalClientName** ADM element.

AccountType = USER_WORKSTATION_TRUST_ACCOUNT as specified in [MS-SAMR].

DesiredAccess = GENERIC_ALL

Upon success, execution MUST continue at step 6.

If an error is returned indicating that the account already exists, execution MUST continue at step 5.

Any other error will fail the task.

5. This processing step MUST be executed if it was determined in the previous call that client account already exists.

   1. The client MUST obtain the account RID of the existing account by calling the SamrLookupNamesInDomain ([MS-SAMR] section 3.1.5.11.2) method specifying the following parameter values:

      Names = **TaskLocalClientName** ADM element.

      Count = 1

   2. The client MUST call the **SamrOpenUser** ([MS-SAMR] section 3.1.5.1.9) method specifying the following parameter values:

      DesiredAccess = GENERIC_ALL

      UserId = RID obtained in the previous call.

      Upon successful return, the UserHandle parameter contains the handle to the existing computer account.

      Any error MUST be treated as a failure. This call sequence MUST be abandoned and step 7 (Closing Handles) as follows MUST be invoked before exiting this step.

6. The client MUST populate TaskLocalNewPassword with a cryptographically strong random value. The password is treated as a **directory string**, even if the actual values generated do not map to actual glyphs in the Unicode character set.<12>

7. The client MUST set the new client account password by calling the **SamrSetInformationUser2** method ([MS-SAMR] section 3.1.5.6.4) specifying the following parameters:

   UserInformationClass, Buffer = **TaskLocalPassword** task ADM element MUST be present in any information buffer used.

8. Regardless of whether an error was encountered in any of the preceding calls, any SAM RPC domain controller handles opened MUST be closed using **SamrCloseHandle** method ([MS-SAMR] section 3.1.5.13.1).

Upon successful completion of the preceding call sequence, the client has successfully created or updated the client account in the domain. The following task ADM elements are updated as well:

TaskLocalPassword

Copyright © 2011 Microsoft Corporation.

Any failure in any of the calls in this step is treated as a failure of this step. If this step fails, the task MUST start executing the rollback steps in section 7.4.5.8, and the task MUST fail.

7.4.5.5 Update Client Computer Account

The client follows these steps to update account attributes.<13>

1. The client MUST bind to the DRS RPC endpoint ([MS-DRDM] section 2.1) on **TaskLocalDomainController**.
2. The client MUST invoke the **IDL_DRSCrackNames** method ([MS-DRDM] section 4.1.3) with the following parameter values:

   rpNames = **TaskLocalDomainName.NetBIOS** "\" **TaskLocalClientName**
3. Upon success, the client MUST update the following ADM elements:

   TaskLocalClientAccountDN = Distinguished Name (DN) of the client account returned in the preceding call.
4. The client MUST connect to the LDAP service on **TaskLocalDomainController** as per LDAP specification [RFC2251], and MUST perform an LDAP bind to authenticate the connection using **TaskLocalClientName** and **TaskLocalPassword** as credentials. using the process described in [MS-ADTS] section 5.1.1.<14>
5. The client SHOULD update the following attributes on the client account with the indicated values, using the LDAP connection created in step 4.<15>

Attributes	Values
serverPrincipalName ([MS-ADA3] section 2.252)	"host/**ComputerName.NetBIOS**", "host/**ComputerName.NetBIOS**"."**TaskLocalDomainName.FQDN**", "RestrictedKrbHost/**ComputerName.NetBIOS**", "RestrictedKrbHost/**ComputerName.NetBIOS**"."**TaskLocalDomainName.FQDN**".
dNSHostName ([MS-ADA1] section 2.185)	Directory string value set to "**TaskLocalDomainName.FQDN**"."**TaskLocalDomainName.FQDN**"

6. The client MUST unbind from the LDAP service as per LDAP specification [RFC2251]. This MUST be performed even if errors were encountered in the preceding call sequence. The errors in this operation are ignored.

If this step fails, the task MUST start executing the rollback steps in section 7.4.5.8, and the task MUST fail.

7.4.5.6 Enumerate Domain Trusts

The client MUST enumerate domain trusts by using the steps specified in section 6.4.5.5. If this step fails, the task MUST start executing the rollback steps in section 7.4.5.8, and the task MUST fail.

7.4.5.7 Update Local State

The client MUST update its local state using the steps specified in section 6.4.5.6.

Copyright © 2011 Microsoft Corporation.

If this step fails, the task MUST execute the rollback steps in section 7.4.5.8, and the task MUST fail. If this step succeeds, task execution continues at section 7.4.5.9.

7.4.5.8 Disable New Computer Account on Domain Controller

This step is only executed upon failure of one of the previous steps.

If the task actually created the computer account on the DC (this would be the case if the **SamrCreateUser2InDomain** call succeeded in step 4 of section 7.4.5.4), the task MUST attempt to disable that computer account on the DC using the following steps.

1. The client MUST connect to the SAM RPC server on the domain controller using one of the **SamrConnect** variants. See [MS-SAMR] section 1.7.2 for information about invoking the **SamrConnect** variants in order to the determine version and method supported by the server. See [MS-SAMR] section 3.1.5.1 for using the Open pattern in the SAM interface.

   ServerName = **TaskLocalDomainController.FQDN**

   DesiredAccess = SAM_SERVER_CONNECT | SAM_SERVER_LOOKUP_DOMAIN

2. The client MUST call SamrOpenDomain ([MS-SAMR] section 3.1.5.1.5) specifying the following parameters:

   DesiredAccess = DOMAIN_LOOKUP

   DomainSid = **TaskLocalDomainSID**

3. The client MUST obtain the account RID of the new computer account by calling the **SamrLookupNamesInDomain** ([MS-SAMR] section 3.1.5.11.2) method specifying the following parameter values:

   Names = **TaskLocalClientName**

   Count = 1

4. The client MUST call the SamrOpenUser ([MS-SAMR] section 3.1.5.1.9) method specifying the following parameter values:

   DesiredAccess = USER_FORCE_PASSWORD_CHANGE | USER_READ_ACCOUNT | USER_WRITE_ACCOUNT

   UserId = RID obtained in the previous call.

   If the **SamrOpenUser** call fails, the client MUST retry the call specifying the following parameter values:

   DesiredAccess = USER_FORCE_PASSWORD_CHANGE | USER_READ_ACCOUNT

   UserId = RID obtained in the previous call.

5. The client MUST query the current value of **UserAccountControl** on the computer account using the SamrQueryInformationUser method ([MS-SAMR] section 3.1.5.5.6), specifying the following parameters:

   UserHandle = UserHandle returned in the previous call

   UserInformationClass = **UserControlInformation** (see [MS-SAMR] section 2.2.7.28)

Copyright © 2011 Microsoft Corporation.

6. If the USER_ACCOUNT_DISABLED bit is not set in the **UserAccountControl** value returned by the previous call, the client MUST disable the account using the **SamrSetInformationUser2** method ([MS-SAMR]section 3.1.5.6.4), specifying the following parameters:

   UserHandle = UserHandle returned in step 4

   UserInformationClass = **UserControlInformation** (see [MS-SAMR] section 2.2.7.28)

   Control.UserAccountControl = (value of **UserAccountControl** retrieved in the previous call) | USER_ACCOUNT_DISABLED

7. The client MUST close the previous handles using the **SamrCloseHandle** method ([MS-SAMR] section 3.1.5.13.1) in the following order:

   1. Close the user handle opened in step 4.

   2. Close the domain handle opened in step 2.

   3. Close the server handle opened in step 1.

7.4.5.9 Close Connections

If an SMB/CIFS session was previously established (see section 7.4.5.2), the client MUST disconnect as described in [MS-CIFS], section 3.2.4.24 specifying **TaskLocalSMBSession**.

7.4.5.10 Reinitialize Local Protocols

If the task succeeded, the client MUST reinitialize local protocols using the steps specified in section 6.4.5.8.

7.5 Task Security

Please refer to the Security section of this specification and the Security sections of the referenced protocol Technical Documents (TDs).

Upon successful completion of this task by a client already joined to a domain, the client's machine account in the old domain will be left as it is. The administrator of the old domain should consider disabling or deleting the old machine account as a security best practice.

Copyright © 2011 Microsoft Corporation.

8 Joining a Domain by Creating an Account via LDAP

This section describes the process of joining a client computer to a domain by creating an account via Lightweight Directory Access Protocol (LDAP). This is a secure way of joining the domain. This task shares many of the other actions in establishing the relationship between the client and the domain controller with the task in section 6.

8.1 Task Overview

8.1.1 Task Purpose

The purpose of this task is to securely join a client computer to a domain by creating an account for the client computer as part of the domain join via the LDAP interface.

8.1.2 Task Applicability

This task is applicable to a client attempting to join a domain securely using the LDAP interface. The credentials of a Domain Administrator are required to perform this task.

8.1.3 Task Use Cases

8.1.3.1 Stakeholders and Interests Summary

Client Administrator: The client administrator is the administrator of the client computer, interested in joining the client computer to the domain.

Client Computer: The client computer that is being joined to the domain.

Domain Administrator: The domain administrator is the administrator of the domain. The domain administrator supplies the credentials to this task for creating and configuring the machine account on the domain controller. After the join is completed, the domain administrator is able to influence the client computer via other protocols; see sections 3.1 and 3.4, and [MS-WSO], for further details.

Domain Controller: The domain controller is a computer providing domain services to domain clients. The client creates and configures a machine account on the domain controller during task processing.

8.1.3.2 Supporting Actors and Task Interests Summary

This task depends on the following supporting actors for the specified interests:

- [MS-NRPC] protocol, for locating a domain controller (DC), for establishing a Netlogon binding to a DC and for enumerating the trusted domains.
- [RFC2251] protocol, for creating the client computer account on a DC and for modifying its attributes.

There are no other systems or tasks in which this task is an actor.

Copyright © 2011 Microsoft Corporation.

8.1.3.3 Use Case Diagrams

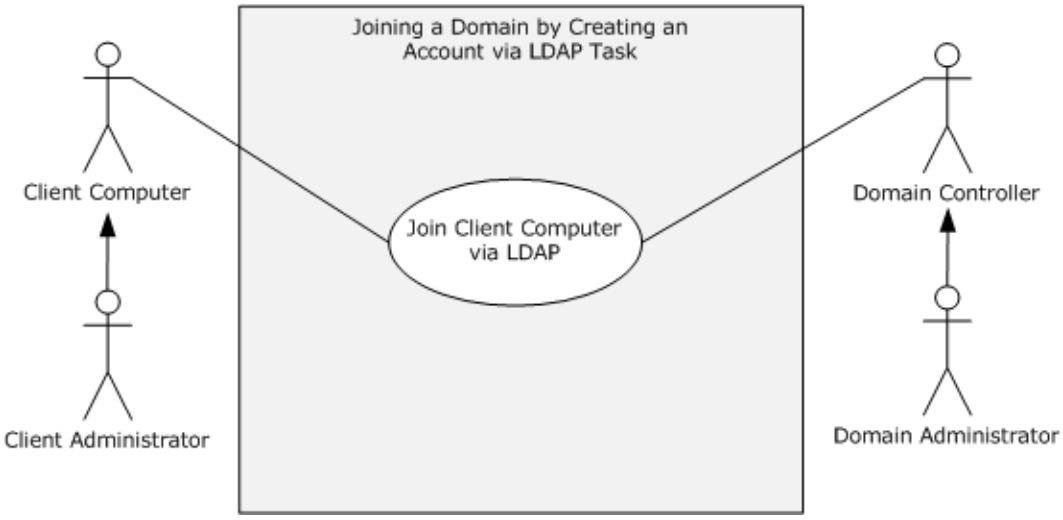

Figure 35: Use case diagram; join a domain by creating an account via LDAP

8.1.3.4 Join a Client Computer to a Domain by Creating an Account via LDAP — Client Computer

The only use case for this task is a client administrator joining the client computer to a domain by providing domain administrator's credentials. The client creates an account for the client computer in the domain via LDAP using domain administrator's credentials.

Goal: Join a Client Computer to a Domain by creating an account for the client computer in the domain via LDAP.

Context of Use: This task is invoked by the client computer administrator in order to enable the client computer to access the services and resources in a domain as well as provide the domain members access to the client computer. See sections 3.1 and 3.4 for other motivations for joining a domain.

Primary Actor: The Client Administrator is the primary actor who initiates joining the client computer to a domain. The Client Administrator invokes commands/applications on the client that initiate the current task.

Direct Actor: The Client Computer joining the domain.

Supporting Actors: All supporting actors are specified in section 8.1.3.2.

Stakeholders and Interests: There are no other stakeholders for the current task besides those listed under Primary Actor, Direct Actor, and Supporting Actors sections as presented earlier.

Precondition: The preconditions for this use case are the same as those listed for the task in section 8.2.3.

Minimal Guarantees: This use case guarantees that a client is either joined to the domain or not joined to a domain upon completion of this use case.

Success Guarantee: The client is joined to the domain.

Copyright © 2011 Microsoft Corporation.

Trigger: This use case is triggered by the client administrator to join the client to a domain.

Main Success Scenario:

1. The Client Administrator initiates the Joining a Domain task on the client computer.
2. The client locates a domain controller.
3. The client connects to the LDAP server on the domain controller and performs a bind to establish a secure LDAP connection using the Domain Administrator's credentials.
4. The client retrieves domain information.
5. The client uses LDAP to create an account in the domain for itself.
6. The client determines the trusted domains.
7. The client updates the client account in the domain.
8. The client updates the local client state.
9. The client reinitializes local protocols.

Extensions: None.

8.2 Task Context

This section describes the relationship between this Task and its environment.

8.2.1 Task Environment

This task assumes that the client is operating in a domain environment with at least one domain controller running. If not, the join task will fail.

8.2.2 Task Relationships

This task builds on the Common Task Information (see section 4), which is shared with all of the tasks in this document.

8.2.2.1 Black Box Relationship Diagrams

The following diagram shows the system as the client computer joins the domain by creating a new account via the Lightweight Directory Access Protocol (LDAP).

Copyright © 2011 Microsoft Corporation.

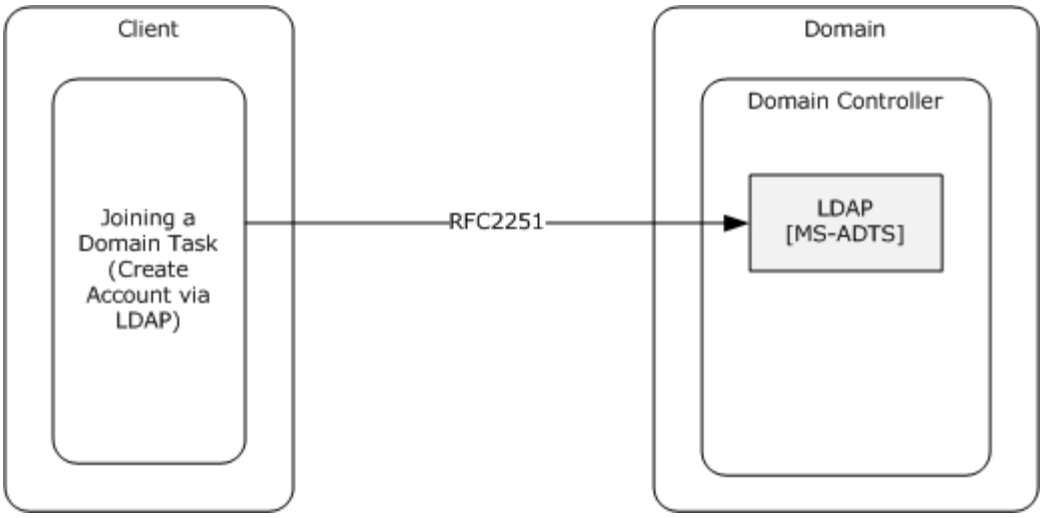

Figure 36: Client machine joining a domain by creating an account via LDAP

8.2.2.2 Task Dependencies

This task depends on the following:

- At least one domain controller being available.
- A domain controller being located using the **DsrGetDcNameEx2** method of the local [MS-NRPC] server.

8.2.2.3 Task Influences

None.

8.2.3 Task Assumptions and Preconditions

The following are the *success conditions* for this task:

- The credentials of an Administrator of the Domain who can create machine accounts in the domain are available to the Client Administrator.

8.2.4 Task Versioning and Capability Negotiation

None.

8.3 Task Architecture

8.3.1 Task Architectural Constraints

This task has the following architectural constraints:

- The client computer can be joined to only a single domain at a time.
- Only one instance of this task MUST run on the client computer at any given time. As the task aims to join the client computer to a single domain, it does not support multiple, parallel executions.

Copyright © 2011 Microsoft Corporation.

- This task MAY be run a second time following a previous successful completion, in order to join the client computer to a different domain.
- This task MAY be run a second time following a previous *unsuccessful* run, as an unsuccessful run makes no changes to the client state.
- This task MUST NOT make assumptions about distributed state, such as the machine account on the domain controller.

8.3.2 Task Abstract Data Model

This section describes state established, used, and maintained by processing rules of this Task. State may be volatile or persisted. State may pertain to one, some, or all instances of the Task. The Task's state consists of the values of the named data elements (also called state variables) presented in this section. The overall organization of the data elements, with their names, is the Abstract Data Model. It is intended to facilitate the reader's conceptual understanding of the specification. While a Task's processing rules may depend upon associations established by the structure of its Abstract Data Model, such association can be achieved in other ways. Implementations may depart from this model so long as their external behavior remains consistent with that described in this document.

The following are the in-memory values used by the Task. These values are not persisted.

Name	Type	Description
TaskLocalClientName	string (Unicode)	Contains the **samAccountName** attribute value for the account in the domain.
TaskLocalPassword	string (Unicode)	Contains the password for the account in the domain.
TaskLocalDomainName.FQDN	string (Unicode)	Contains the FQDN(1) of the domain being joined.
TaskLocalDomainName.NetBIOS	string (Unicode)	Contains the NetBIOS name of the domain being joined.
TaskLocalForestNameFQDN	string (Unicode)	Contains the FQDN(1) of the forest containing the domain being joined.
TaskLocalDomainController	string (Unicode)	Contains the name of the domain controller used by this task.
TaskLocalDomainSID	SID	Contains the SID of the domain being joined.
TaskLocalDomainGUID	GUID	Contains the GUID of the domain being joined.
TaskLocalTrustedDomains	Domains	Contains a list of trusted domains for the domain being joined.
TaskLocalClientAccountDN	string (Unicode)	Contains the Distinguished Name of the client account in the domain.
TaskLocalAlreadyJoined	Boolean	Whether the client is already joined to a domain at the start of the task.

Copyright © 2011 Microsoft Corporation.

Name	Type	Description
TaskLocalLDAPConnection	ADCONNECTION_HANDLE	An ADCONNECTION_HANDLE that refers to an instance of the ADConnection ADM element specified in [MS-ADSO] section 6.2.3.

The following client data model variables (section 4.3.1.1) are updated by the task during a successful completion:

- **DomainSid**
- **DomainGuid**
- **DomainName** (NetBIOS and FQDN Names)
- **ClientName**
- **Password**
- **TrustedDomains**

8.3.3 Task Abstract Parameters

This section describes data passed to an instance of this task at the time it is invoked or triggered. The parameters consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Parameter. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules could depend upon associations established by the structure of its Abstract Parameters, such association can be achieved in other ways. Implementations can depart from this abstraction so long as their external behavior remains consistent with that described in this document.

The parameters to this task are as follows:

Name	Type	Description	Optional
TaskInputDomainName	string (Unicode)	Specifies the NetBIOS or FQDN name of the domain to join.	No
TaskInputDomainController	string (Unicode)	Specifies the domain controller this task must use to complete the join.	Yes
TaskInputDomainAdminAccount	string (Unicode)	Specifies the name of a Domain Administrator account.	No
TaskInputDomainAdminPassword	string (Unicode)	Specifies the password for the *TaskInputDomainAdminAccount*.	No
TaskInputAccountOU	string (Unicode)	Specifies the Distinguished Name (DN) of the OU to create the computer account in.	Yes

8.3.4 Task Abstract Results

This section describes data returned by an instance of this task to its caller. The results consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Result. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules might depend upon associations

Copyright © 2011 Microsoft Corporation.

established by the structure of its Abstract Results, such association can be achieved in other ways. Implementations can depart from this abstraction so long as their external behavior remains consistent with that described in this document.

The task returns the following results to the caller:

Name	Type	Description
TaskReturnStatus	integer	This task MUST return 0x00000000 on success. Error statuses generated by a failure during task processing are in the Win32 error space (a long data type), as specified in [MS-ERREF] section 2.2.

Upon a successful task completion this task performs the following:

1. Updates client ADM variables (section 4.3.1.1). The updated values are persisted, as specified in section 4.3.1.1, and are available for other protocols on the client to use.
2. Starts the **NetLogon** Remote Protocol and configures it to run automatically at every system start up.
3. Notifies the Certificate Autoenrollment system that the machine is joining the domain.
4. Adds the Domain Administrators to the local Administrators group (see invocation of [MS-SAMR] section 3.1.7.1).
5. Notifies the local MS-SNTP protocol that the machine is joining the domain (failures in this step are ignored).
6. Creates or modifies a client computer account in the domain.

Upon a failed task completion, this task will attempt to delete the computer account in the domain (only if the task created the account, as opposed to modifying an existing account), and will attempt to revert any local state changes as well. If the task is unable to disable the computer account upon failure, domain administrator intervention outside the scope of the task may be necessary to delete the account. If the task is unable to revert local state changes, administrator intervention outside the scope of the task may be necessary to reconfigure the client back to its original state.

8.3.5 White-Box Relationships

The following diagram represents the white-box relationships of the task within the client computer and with the domain controller.

Copyright © 2011 Microsoft Corporation.

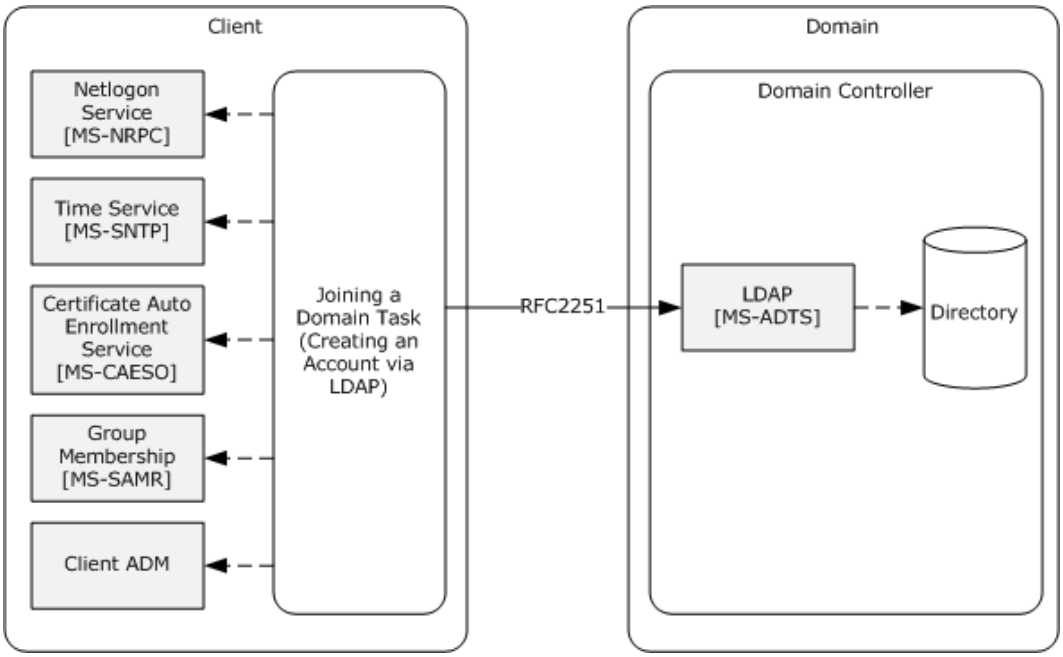

Figure 37: Task white-box relationships diagram

8.3.6 Task Events

8.3.6.1 Task Timers

None.

8.3.6.2 Task Non-Timer Events

None.

8.3.7 Task Architecture and Communication

The following details document boxes in the diagram as shown:

Joining a Domain Task (Creating an Account via LDAP): This box represents the task, including the task ADM and processing (see section 8.4.5). This box invokes the local [MS-NRPC] server to locate a domain controller. This box interacts with the domain controller for achieving the goals of this task. This box also configures client ADM, several client local states and fires required triggers as a final step in a successful run.

Client ADM, Group Membership: These boxes are configured by the Joining a Domain Task box as one of the final steps on a successful run of the task.

Time Service, Cert Auto Enrollment svc: These boxes are signaled by the Joining a Domain Task box as one of the final steps on a successful run of the task.

Netlogon Service: This box is configured by the Joining a Domain Task box as one of the final steps on a successful run of the task.

LDAP: This box represents the server-side implementations of LDAP.

Copyright © 2011 Microsoft Corporation.

Directory: This box represents the Directory Service running on the Domain Controller and it holds the newly created account for the client computer.

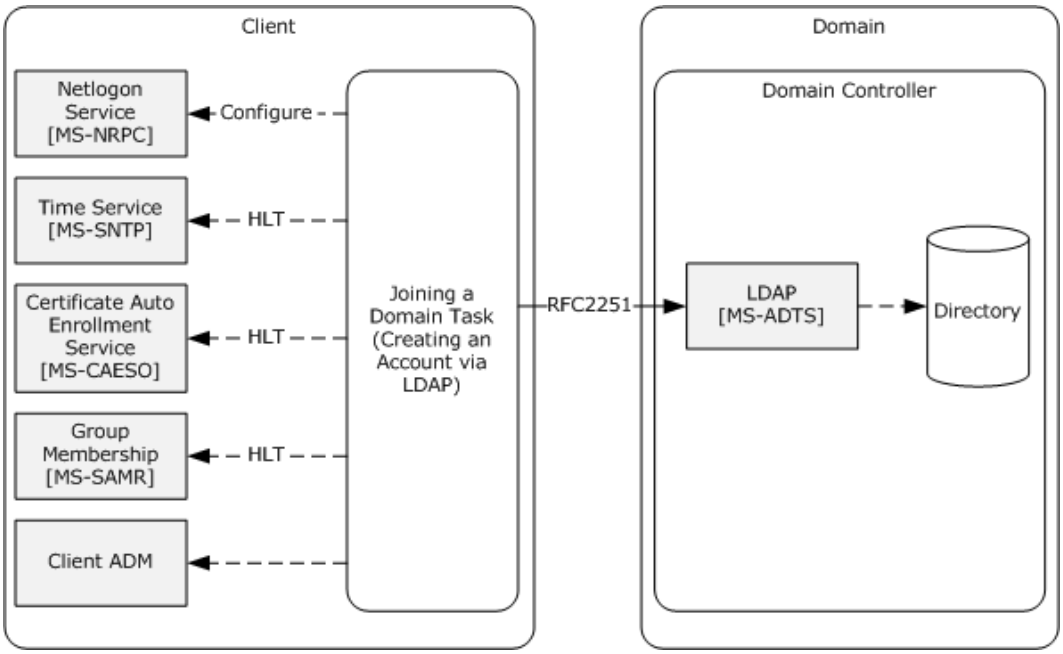

Figure 38: Client joining a domain (Creating an Account via LDAP) – architecture and communication

8.3.8 Task Processing Rules

Abstract Parameters: As specified in section 8.3.3.

Main Success Scenario:

1. Locate a domain controller (DC) to be used during task processing (see section 8.4.5.1).
2. Establish an LDAP connection to the domain controller (see section 8.4.5.2).
3. Retrieve domain information from the domain controller (see section 8.4.5.3).
4. Create client computer account on the domain controller (see section 8.4.5.4).
5. Enumerate domain trusts (see section 8.4.5.5).
6. Update local state (see section 8.4.5.6).
7. Close connections (see section 8.4.5.8).
8. Reinitialize local protocols (see section 8.4.5.9).

Extensions:

1. Unable to locate a domain controller, terminate task with error.
2. Unable to establish to the LDAP interface to the domain controller, terminate task with error.

Copyright © 2011 Microsoft Corporation.

3. Domain information cannot be retrieved, terminate task with error.
4. Computer account cannot be created, terminate task with error.
5. Domain trust enumeration fails, rollback changes on the DC (section 8.4.5.7) and terminate task with error.
6. Update local state fails, rollback changes on the DC (section 8.4.5.7) and terminate task with error.

8.3.9 Task Failure Scenarios

The following are the common failure scenarios for the current task:

- **Unable to Locate DC**: If a domain controller in the domain identified by **TaskInputDomainName** cannot be located, (step 1 in Main Success Scenario, section 8.3.8). The result is that the join task fails. This condition MAY occur if the network infrastructure is not configured correctly or is unavailable.
- **Domain Controller Unavailable**: If **TaskLocalDomainController** cannot be reached due to any reason while the client opens connections or queries or updates the domain controller (all the domain controller interactions in Main Success Scenario, section 8.3.8), the task fails.
- **Authentication Fails**: If the **TaskInputDomainAdminAccount** and derived **TaskInputDomainAdminPassword** do not authenticate the Client binding to the server (step 2 in Main Success Scenario, section 8.3.8), further processing is not possible and the task fails.
- **Queries or Updates fail on domain controller**: If any of the queries or updates made to **TaskLocalDomainController** fails (in steps 2 through 5 in Main Success Scenario, section 8.3.8), the task fails.
- **Update fails on client**: Update to client local state fails (6) and as a result the join task fails. This failure MAY happen because of conditions on the client computer.

8.4 Task Details

This section contains the details that complete the descriptions in earlier sections of the document. These are needed to understand and implement this Task.

8.4.1 Task Precondition Details

Not applicable.

8.4.2 Task Initialization of External Entities

Once a domain controller is located using the Locate Domain Controller task, communication is directly with the domain controller.

8.4.3 Task Event Details

8.4.3.1 Task Timer Details

None.

Copyright © 2011 Microsoft Corporation.

8.4.3.2 Task Non-Timer Event Details

None.

8.4.4 Task Architectural Details

This task is a variant of the Joining a Domain by Creating Account via SAMR task (section 7). In this task, the client computer uses the LDAP protocol for connecting to, querying and updating the domain controller.

The client determines a domain controller in the domain it is joining using the DsrGetDcNameEx2 method of the local [MS-NRPC] server. The client then binds to the domain controller's LDAP interface and opens a secure connection, using the Domain Administrator's credentials for authentication. The client uses the established connection for the subsequent queries and updates to the domain controller.

The client retrieves the necessary domain information, then creates the client computer account on the domain controller. The client then retrieves domain trusts from the domain controller. The client then updates SPN list for the client computer account on the domain controller. Finally, the client updates its local state.

The following figure shows the network traffic for a typical main success scenario, where **TaskInputAccountDN** was not specified and the client account does not already exist on the domain controller. Note that the initial exchange (Locate a DC) is representative only of the traffic between the client and the selected domain controller; additional exchanges that may occur to other domain controllers are not represented. See section 5.4.4 for additional details.

Copyright © 2011 Microsoft Corporation.

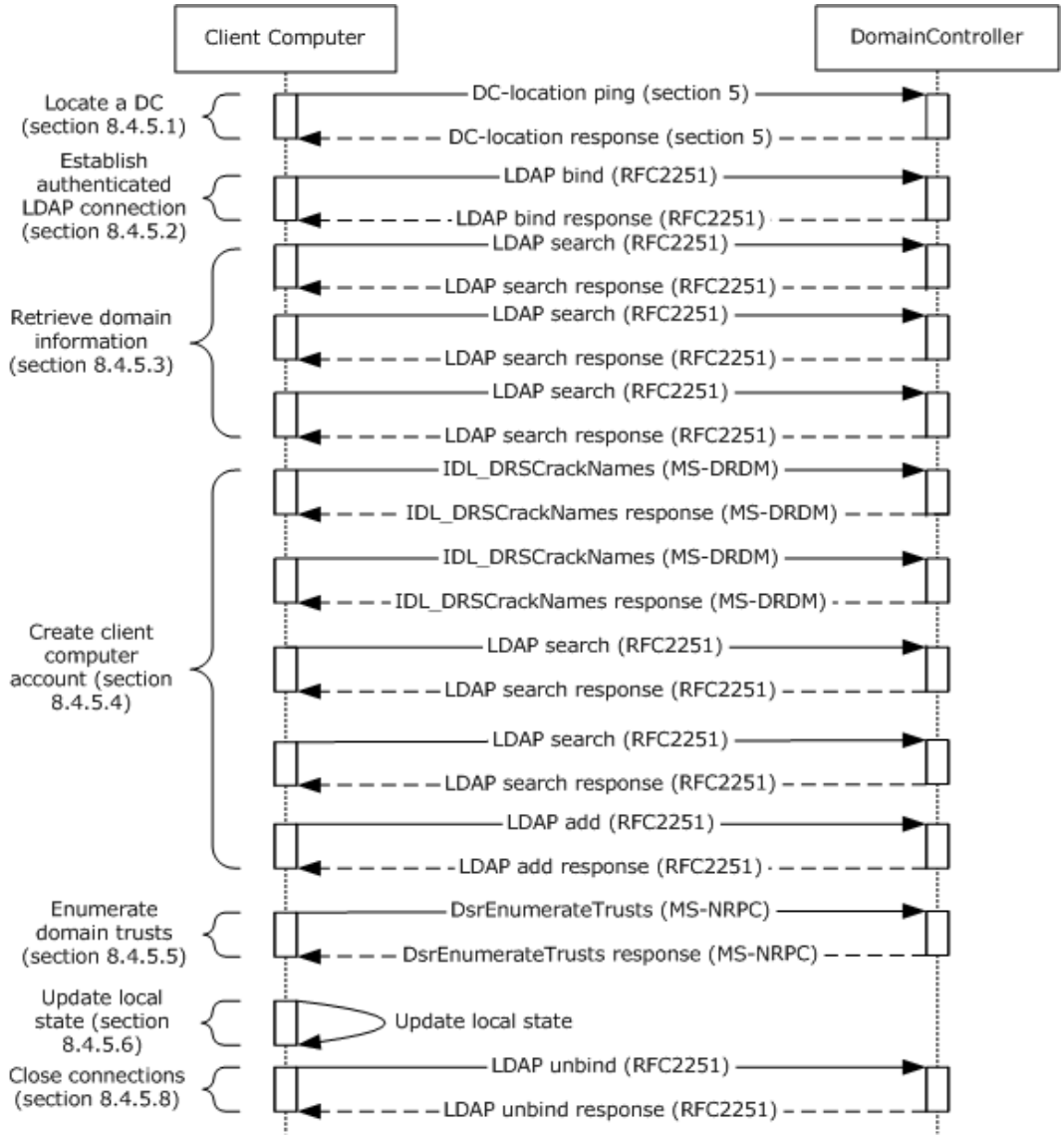

Figure 39: Joining a Domain (Creating account via LDAP) task architectural details

8.4.5 Task Processing Rule Details

This section describes details for the steps identified in section 8.3.8. Unless otherwise specified, the processing falls through from one section to the next.

The following flowchart for the current task shows the sequential steps in the task, as well as failure decisions taken based on the results in each step.

Copyright © 2011 Microsoft Corporation.

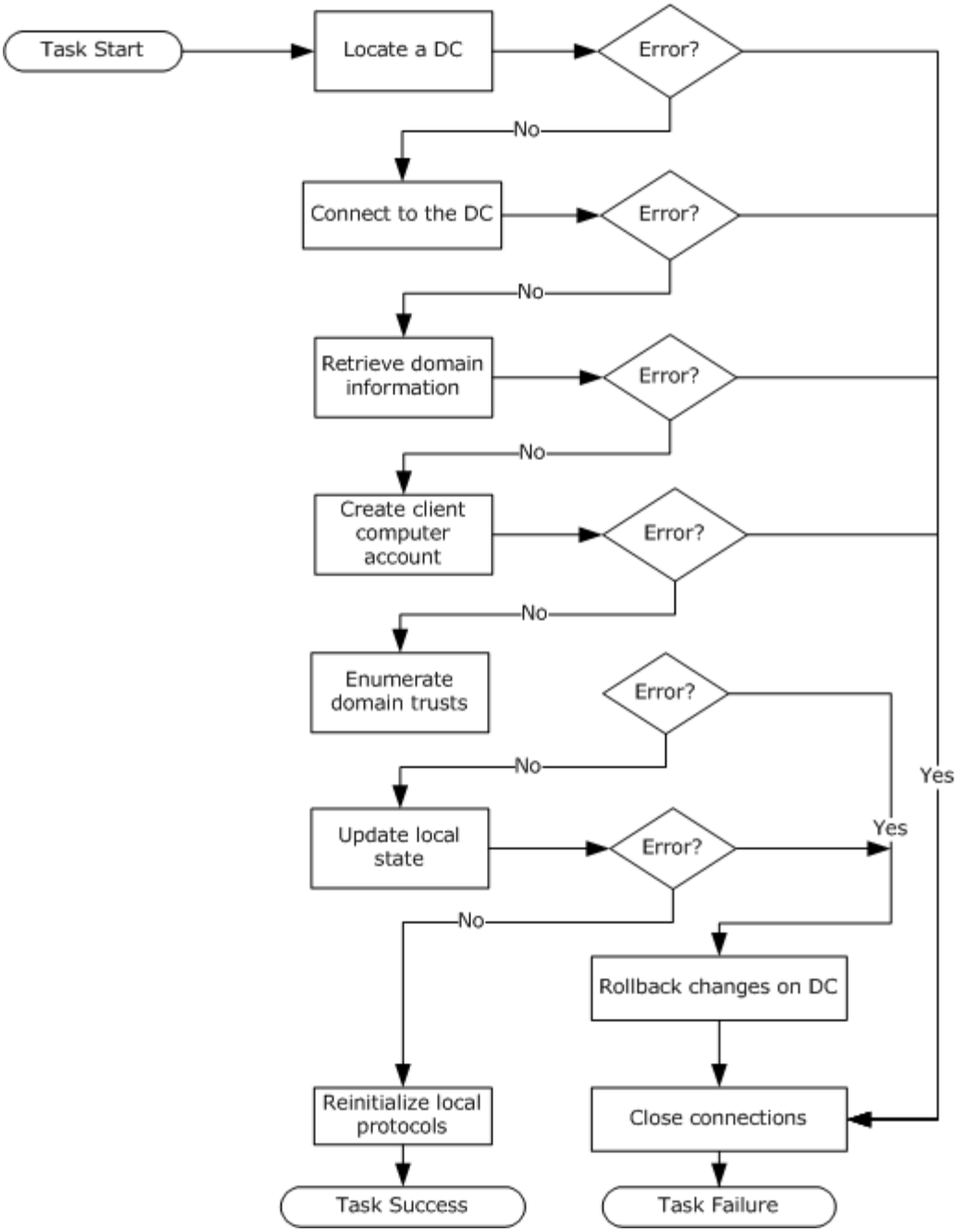

Figure 40: Joining a Domain (Creating account via LDAP) Task flowchart

The abstract parameters specified in section 8.3.3 are passed to this task upon entry.

The initial state of the in-memory task ADM is initialized as follows:

```
TaskLocalClientName = ComputerName.NetBIOS "$"
TaskLocalPassword = <undefined>
TaskLocalDomainSID = <undefined>
```


Copyright © 2011 Microsoft Corporation.

```
TaskLocalDomainGUID = <undefined>
TaskLocalDomainController = TaskInputDomainController
TaskLocalDomainName = <undefined>
TaskLocalForestNameFQDN = <undefined>
TaskLocalClientAccountDN = <undefined>
TaskLocalAlreadyJoined = FALSE if DomainSid is equal to NULL, TRUE otherwise
```

8.4.5.1 Locate a Domain Controller

This step of the task processing is the same as section 6.4.5.1, with the same actions on success and failure outcomes as described in that section.

8.4.5.2 Establish Authenticated LDAP Connection

1. The client invokes the "Initializing an ADConnection" task ([MS-ADSO] section 6.2.6.1.1) with the following parameters:

   - *TaskInputTargetName*: **TaskLocalDomainController**
   - *TaskInputPortNumber*: 389

   Upon success, the result is stored in **TaskLocalLDAPConnection**.

2. The client invokes the "Setting an LDAP option on an ADConnection" task ([MS-ADSO] section 6.2.6.1.2) with the following parameters:

   - *TaskInputADConnection*: NewADConnection
   - *TaskInputOptionName*: LDAP_OPT_AUTH_INFO
   - *TaskInputOptionValue*:
     - bindMethod: SASL, using the GSS-SPNEGO protocol ([MS-ADTS] section 5.1.1.1)
     - name: **TaskInputDomainAdminAccount**
     - password: **TaskInputDomainAdminAccountPassword**

3. The client invokes the "Setting an LDAP option on an ADConnection" task ([MS-ADSO] section 6.2.6.1.2) with the following parameters:

   - *TaskInputADConnection*: TaskLocalLDAPConnection
   - *TaskInputOptionName*: LDAP_OPT_AREC_EXCLUSIVE
   - *TaskInputOptionValue*: TRUE

4. The client invokes the "Setting an LDAP option on an ADConnection" task ([MS-ADSO] section 6.2.6.1.2) with the following parameters:

   - *TaskInputADConnection*: TaskLocalLDAPConnection
   - *TaskInputOptionName*: LDAP_OPT_ENCRYPT
   - *TaskInputOptionValue*: TRUE

Copyright © 2011 Microsoft Corporation.

5. The client invokes the "Setting an LDAP option on an ADConnection" task ([MS-ADSO] section 6.2.6.1.2) with the following parameters:
   - *TaskInputADConnection*: TaskLocalLDAPConnection
   - *TaskInputOptionName*: LDAP_OPT_REFERRALS
   - *TaskInputOptionValue*: FALSE
6. The client invokes the "Establishing an ADConnection" task ([MS-ADSO] section 6.2.6.1.3) with the *TaskInputADConnection* parameter set to TaskLocalLDAPConnection.
7. The client invokes the "Performing an LDAP Bind on an ADConnection" task ([MS-ADSO] section 6.2.6.1.4) with the *TaskInputADConnection* parameter set to NewADConnection.

8.4.5.3 Retrieve Domain Information

1. The client invokes the "Performing an LDAP Operation on an ADConnection" task ([MS-ADSO] section 6.2.6.1.6) with the following parameters:
   - *TaskInputADConnection*: **TaskLocalLDAPConnection**
   - *TaskInputRequestMessage*: LDAP SearchRequest message [RFC2251] section 4.5.1 as follows:
     - baseObject: Null DN
     - scope: baseObject
     - filter: ObjectClass=*
     - attributes: defaultNamingContext, rootDomainNamingContext, configurationNamingContext, supportedControl
     - derefAliases: neverDerefAliases
     - typesOnly: false
   - *TaskOutputResultMessages*: LDAPResultMessages

   Upon success, the client MUST set the following task ADM elements:

   TaskLocalDomainName.FQDN = (value of defaultNamingContext reformatted as an FQDN)

   TaskLocalForestName.FQDN = (value of rootDomainNamingContext reformatted as an FQDN)
2. The client invokes the "Performing an LDAP Operation on an ADConnection" task ([MS-ADSO] section 6.2.6.1.6) with the following parameters:
   - *TaskInputADConnection*: **TaskLocalLDAPConnection**
   - *TaskInputRequestMessage*: LDAP SearchRequest message [RFC2251] section 4.5.1 as follows:
     - baseObject: defaultNamingContext that is obtained in step 1
     - scope: baseObject
     - filter: ObjectClass=*
     - attributes: objectGuid, objectSid

Copyright © 2011 Microsoft Corporation.

- derefAliases: neverDerefAliases
- typesOnly: false

- *TaskOutputResultMessages*: LDAPResultMessages

Upon success, the client MUST set the following task ADM elements:

TaskLocalDomainGUID = objectGuid

TaskLocalDomainSID = objectSid

3. The client invokes the "Performing an LDAP Operation on an ADConnection" task ([MS-ADSO] section 6.2.6.1.6) with the following parameters:

- *TaskInputADConnection*: **TaskLocalLDAPConnection**
- *TaskInputRequestMessage*: LDAP SearchRequest message [RFC2251] section 4.5.1 as follows:
  - baseObject: concatenation of "CN=Partitions," with the value of configurationNamingContext obtained in step 1
  - scope: wholeSubtree
  - filter: string"(&(objectClass=crossref)(ncName=defaultNamingContext)", where defaultNamingContext represents the value obtained in step 1
  - attributes: nETBIOSName
  - derefAliases: neverDerefAliases
  - typesOnly: false
- *TaskOutputResultMessages*: LDAPResultMessages

Upon success, the client MUST set the following task ADM element:

TaskLocalDomainName.NetBios = nETBIOSName

8.4.5.4 Create Client Computer Account on the Domain Controller

The client MUST create an account in the directory to represent itself, or modify an existing account as required. This is achieved as follows:

1. The client MUST bind to the DRS RPC endpoint ([MS-DRDM] section 2.1) on **TaskLocalDomainController**.
2. The client MUST invoke IDL_DRSCrackNames Method ([MS-DRDM] section 4.1.3) with the following parameter values:

   rpNames = **TaskLocalDomainName.NetBIOS** "\" **TaskLocalClientName**

   If the call succeeds, **TaskInputAccountOU** was specified, and the account exists under **TaskInputAccountOU**, the client MUST set **TaskLocalClientAccountDN** as follows:

   TaskLocalClientAccountDN = **TaskInputAccountOU** "," **TaskLocalClientName**

   Execution then continues at step 6.

Copyright © 2011 Microsoft Corporation.

If the call succeeds, **TaskInputAccountOU** was specified, and the account does not exist, the client MUST continue execution at step 3.

If the call succeeds, **TaskInputAccountOU** was specified, and the account exists under a different OU than **TaskInputAccountOU**, the client MUST fail the task.

If the call succeeds, **TaskInputAccountOU** was specified, and the account does not exist, the client MUST set **TaskLocalClientAccountDN** as follows:

TaskLocalClientAccountDN = **TaskInputAccountOU** "," **TaskLocalClientName**

Execution then continues at step 6.

If the call succeeds, **TaskInputAccountOU** was not specified, and the account does not exist, the client MUST continue execution at step 4.

If the call fails, the client MUST fail the task.

3. The client MUST validate that the requested OU (**TaskInputAccountOU**) is valid for use. The client invokes the "Performing an LDAP Operation on an ADConnection" task ([MS-ADSO] section 6.2.6.1.6) with the following parameters:

   - *TaskInputADConnection*: **TaskLocalLDAPConnection**
   - *TaskInputRequestMessage*: LDAP CompareRequest message [RFC2251] section 4.10, as follows:
     - entry: **TaskInputAccountOU**
     - ava: **OrganizationalUnit**
   - *TaskOutputResultMessages*: LDAPResultMessages

   If the call returns LDAP_COMPARE_TRUE, the requested OU is validated and the client MUST set **TaskLocalClientAccountDN** as follows:

   TaskLocalClientAccountDN = **TaskInputAccountOU** "," **TaskLocalClientName**

   Execution then continues at step 6.

   If the call fails or returns LDAP_COMPARE_FALSE, the task MUST fail.

4. The client MUST retrieve the default OU for computer accounts from the DC to determine where to create the new computer account.

   The client invokes the "Performing an LDAP Operation on an ADConnection" task ([MS-ADSO] section 6.2.6.1.6) with the following parameters:

   - *TaskInputADConnection*: **TaskLocalLDAPConnection**
   - *TaskInputRequestMessage*: LDAP SearchRequest message [RFC2251] section 4.5.1 as follows:
     - baseObject: TaskLocalDomainController
     - scope: baseObject
     - filter: ObjectClass=computer
     - attributes: preferredOU

Copyright © 2011 Microsoft Corporation.

- derefAliases: neverDerefAliases
- typesOnly: false

- *TaskOutputResultMessages*: LDAPResultMessages

If the search fails, the task MUST fail. If the search succeeds, the client MUST parse the returned value for the DN of the computer container DN (see [MS-ADTS] section 7.1.1.4 for details) and get the DN of the preferredOU.

5. The client invokes the "Performing an LDAP Operation on an ADConnection" task ([MS-ADSO] section 6.2.6.1.6) with the following parameters:

   - *TaskInputADConnection*: **TaskLocalLDAPConnection**
   - *TaskInputRequestMessage*: LDAP SearchRequest message [RFC2251] section 4.5.1 as follows:
     - baseObject: The DN of preferredOU obtained in step 4.
     - scope: baseObject
     - filter: ObjectClass=*
     - attributes: wellKnownObjects
     - derefAliases: neverDerefAliases
     - typesOnly: false
   - *TaskOutputResultMessages*: LDAPResultMessages

   If the search fails, the task MUST fail. If the search succeeds, the client MUST parse the returned value for the DN of the computer container DN (see [MS-ADTS] section 7.1.1.4 for details) and set **TaskLocalClientAccountDN** as follows:

   TaskLocalClientAccountDN = **WellknownComputerAccountOU** "," **TaskLocalClientName**

6. The client MUST generate a password through cryptographically sound random number generators. The password is treated as a Unicode string, even if the actual values generated do not map to actual glyphs in the Unicode character set. <16>

   The **TaskLocalPassword** ADM element is updated with the newly generated password.

7. The client invokes the "Performing an LDAP Operation on an ADConnection" task ([MS-ADSO] section 6.2.6.1.6) with the following parameters:

   - *TaskInputADConnection*: **TaskLocalLDAPConnection**
   - *TaskInputRequestMessage*: LDAP SearchRequest message [RFC2251] section 4.5.1 as follows:
     - baseObject: **TaskLocalClientAccountDN**
     - scope: baseObject
     - filter: ObjectClass=*
     - attributes: objectClass, dnsHostName, sAMAccountName, servicePrincipalName, userAccountControl

Copyright © 2011 Microsoft Corporation.

- derefAliases: neverDerefAliases
- typesOnly: false

- *TaskOutputResultMessages*: LDAPResultMessages

8. If step 7 fails with an error indicating that the account does not exist, the client invokes the "Performing an LDAP Operation on an ADConnection" task ([MS-ADSO] section 6.2.6.1.6) with the following parameters:

   - *TaskInputADConnection*: **TaskLocalLDAPConnection**
   - *TaskInputRequestMessage*: LDAP AddRequest message [RFC2251] section 4.7 as follows:
     - entry: **TaskLocalClientAccountDN**
     - attributes: AttributeList formed by following attributes and values:

Attribute	Value
objectClass	Set to computer ([MS-ADSC] section 2.21).
dNSHostName	"**ComputerName.NetBIOS**"."**TaskLocalDomainName.FQDN**"
sAMAccountName	**TaskLocalClientName**
servicePrincipalName	host/"**ComputerName.NetBIOS**" host/"**ComputerName.NetBIOS**"."**TaskLocalDomainName.FQDN**" RestrictedKrbHost/"**ComputerName.NetBIOS**" RestrictedKrbHost/"**ComputerName.NetBIOS**"."**TaskLocalDomainName.FQDN**"
userAccountControl	0x1000
unicodePwd	**TaskLocalPassword**

   - TaskOutputResultMessages: LDAPResultMessages

   If the call fails, the client MUST fail the task. Otherwise execution continues at section 8.4.5.5.

9. If step 7 succeeds, the client MUST compare the returned value of each attribute against the desired values listed in the table in step 8.

   The client invokes the "Performing an LDAP Operation on an ADConnection" task ([MS-ADSO] section 6.2.6.1.6) with the following parameters:

   - *TaskInputADConnection*: **TaskLocalLDAPConnection**
   - *TaskInputRequestMessage*: LDAP CompareRequest message [RFC2251] section 4.10 as follows:
     - entry: Set to Attribute from the table in step 8.
     - ava: Set to corresponding Value from the table in step 8.
   - *TaskOutputResultMessages*: LDAPResultMessages

Copyright © 2011 Microsoft Corporation.

If the compare request returns LDAP_COMPARE_TRUE for all the attributes, the client invokes the "Performing an LDAP Operation on an ADConnection" task ([MS-ADSO] section 6.2.6.1.6) with the following parameters:

- *TaskInputADConnection*: **TaskLocalLDAPConnection**
- *TaskInputRequestMessage*: LDAP ModifyRequest message [RFC2251] section 4.6, as follows:
  - object: **TaskLocalClientAccountDN**
  - operation: replace
  - modification type: unicodePwd
  - modification vals: **TaskLocalPassword**
- *TaskOutputResultMessages*: LDAPResultMessages, which indicates whether a modification is successful.

If the compare request returns LDAP_COMPARE_FALSE for any of the attribute and value pairs specified in the table shown in step 8, the client MUST perform an LDAP modify operation against the directory on TaskLocalDomainController to set the attribute and values as specified in the table in step 8. The client invokes the "Performing an LDAP Operation on an ADConnection" task ([MS-ADSO] section 6.2.6.1.6) with the following parameters:

- *TaskInputADConnection*: **TaskLocalLDAPConnection**
- *TaskInputRequestMessage*: LDAP ModifyRequest message [RFC2251] section 4.6, as follows:
  - object: **TaskLocalClientAccountDN**
  - operation: replace
  - modification type: The attribute used in the LDAP compare request that returned LDAP_COMPARE_FALSE.
  - modification vals: The corresponding value specified in the step 8 table for that attribute.
- *TaskOutputResultMessages*: LDAPResultMessages

If the call fails, the client MUST fail the task.

8.4.5.5 Enumerate Domain Trusts

The client MUST enumerate domain trusts using the steps specified in section 6.4.5.5.

If this step fails, the task MUST execute the rollback steps in section 8.4.5.7, and the task MUST fail.

8.4.5.6 Update Local State

The client MUST update its local state using the steps specified in section 6.4.5.6.

If this step fails, the task MUST execute the rollback steps in section 8.4.5.7, and the task MUST fail. If this step succeeds, task execution continues at section 8.4.5.8.

8.4.5.7 Rollback Changes on Domain Controller

This step is only executed upon failure of a previous step.

Copyright © 2011 Microsoft Corporation.

If the task actually created the computer account on the DC (this is true if the LDAP add operation succeeded in step 8 of section 8.4.5.4), the task MUST attempt to delete that computer account on the DC using the steps that follow.

The following definitions are used in the specification of message processing that follows in this section.

- **CleanupLDAPConnection**: an ADCONNECTION_HANDLE, described in [MS-ADSO] section 6.2.3, to a domain controller.

1. The client invokes the "Initializing an ADConnection" task ([MS-ADSO] section 6.2.6.1.1) with the following parameters:

   - *TaskInputTargetName*: **TaskLocalDomainController**
   - *TaskInputPortNumber*: 389

   Upon success, the result is stored in CleanupLDAPConnection.

2. The client invokes the "Setting an LDAP option on an ADConnection" task ([MS-ADSO] section 6.2.6.1.2) with the following parameters:

   - TaskInputADConnection: CleanupLDAPConnection
   - TaskInputOptionName: LDAP_OPT_AUTH_INFO
   - TaskInputOptionValue:
     - bindMethod: SASL, using the GSS-SPNEGO protocol ([MS-ADTS] section 5.1.1.1)
     - name: **TaskInputDomainAdminAccount**
     - Password: **TaskInputDomainAdminAccountPassword**

3. The client invokes the "Setting an LDAP option on an ADConnection" task ([MS-ADSO] section 6.2.6.1.2) with the following parameters:

   - *TaskInputADConnection*: CleanupLDAPConnection
   - *TaskInputOptionName*: LDAP_OPT_AREC_EXCLUSIVE
   - *TaskInputOptionValue*: TRUE

4. The client invokes the "Setting an LDAP option on an ADConnection" task ([MS-ADSO] section 6.2.6.1.2) with the following parameters:

   - *TaskInputADConnection*: CleanupLDAPConnection
   - *TaskInputOptionName*: LDAP_OPT_ENCRYPT
   - *TaskInputOptionValue*: TRUE

5. The client invokes the "Setting an LDAP option on an ADConnection" task ([MS-ADSO] section 6.2.6.1.2) with the following parameters:

   - *TaskInputADConnection*: CleanupLDAPConnection
   - *TaskInputOptionName*: LDAP_OPT_REFERRALS

Copyright © 2011 Microsoft Corporation.

- *TaskInputOptionValue*: FALSE

6. The client invokes the "Establishing an ADConnection" task ([MS-ADSO] section 6.2.6.1.3) with the TaskInputADConnection parameter set to CleanupLDAPConnection.
7. The client invokes the "Performing an LDAP Bind on an ADConnection" task ([MS-ADSO] section 6.2.6.1.4) with the *TaskInputADConnection* parameter set to CleanupLDAPConnection.
8. The client invokes the "Performing an LDAP Operation on an ADConnection" task ([MS-ADSO] section 6.2.6.1.6) with the following parameters:
   - *TaskInputADConnection*: CleanupLDAPConnection
   - *TaskInputRequestMessage*: LDAP modifyRequest message [RFC2251] section 4.6 as follows:
     - Object: **TaskLocalClientAccountDN**
     - The modification sequence has one list entry, set as follows:
       - First list entry
         - operation: delete
         - modification:
           - type: msDS-AdditionalDnsHostName
           - vals: "**ComputerName.NetBIOS**"."**TaskLocalDomainName.FQDN**"
     - controls: Sequence of one Control structure, as follows:
       - controlType: LDAP_SERVER_PERMISSIVE_MODIFY_OID ([MS-ADTS] section 3.1.1.3.4.1.8)
       - criticality: FALSE
   - *TaskOutputResultMessages*: LDAPResultMessages

   This task will return TaskReturnStatus, the LDAP resultCode ([RFC2251] section 4.1.10) returned from the directory server in response to the request, or an error indicating that the directory server could not be contacted or that a timeout has occurred.
9. The client MUST invoke task "Performing an LDAP Unbind on an ADConnection" ([MS-ADSO] section 6.2.6.1.5) with the *TaskInputADConnection* parameter set to CleanupLDAPConnection.

8.4.5.8 Close Connections

The client MUST invoke the task "Performing an LDAP Unbind on an ADConnection", as specified in [MS-ADSO] section 6.2.6.1.5, with the *TaskInputADConnection* parameter set to **TaskLocalLDAPConnection**.

8.4.5.9 Reinitialize Local Protocols

The client MUST reinitialize local protocols using the steps specified in section 6.4.5.8.

Copyright © 2011 Microsoft Corporation.

8.5 Task Security

Please refer to the Security section of this specification and the Security sections of the referenced protocol Technical Documents (TDs).

Upon successful completion of this task by a client already joined to a domain, the client's machine account in the old domain will be left as it is. The administrator of the old domain should consider disabling or deleting the old machine account as a security best practice.

Copyright © 2011 Microsoft Corporation.

9 Unjoining a Domain Member

This section describes the process of unjoining a client computer from a domain.

9.1 Task Overview

9.1.1 Task Purpose

The purpose of this task is to unjoin a client computer from a domain.

9.1.2 Task Applicability

This task is applicable to a client computer that is part of a domain and needs to unjoin from the domain.

9.1.3 Task Use Cases

9.1.3.1 Stakeholders and Interests Summary

Client Administrator: The Client Administrator is the administrator of the client computer, interested in unjoining the client computer from the domain.

Client Computer: The client computer that is being unjoined from the domain.

Domain Administrator: The Domain Administrator is the administrator of the domain. The Domain Administrator supplies the credentials to this task for disabling the machine account on the domain controller (DC).

Domain Controller: The domain controller (DC) is a computer providing domain services to domain clients. The Domain Administrator disables the machine account on the domain controller during task processing.

9.1.3.2 Supporting Actors and Task Interests Summary

This task depends on the following supporting actors for the specified interests:

- [MS-NRPC] local protocol, for locating a local a domain controller.
- SMB protocol ([MS-SMB2], [MS-SMB], or [MS-CIFS]), for opening/closing SMB sessions to a DC. The Task does not require that any specific SMB protocol be used.
- [MS-SAMR] protocol, for disabling the client computer account on a DC.

There are no other systems or Tasks in which this Task is an actor.

Copyright © 2011 Microsoft Corporation.

9.1.3.3 Use Case Diagram

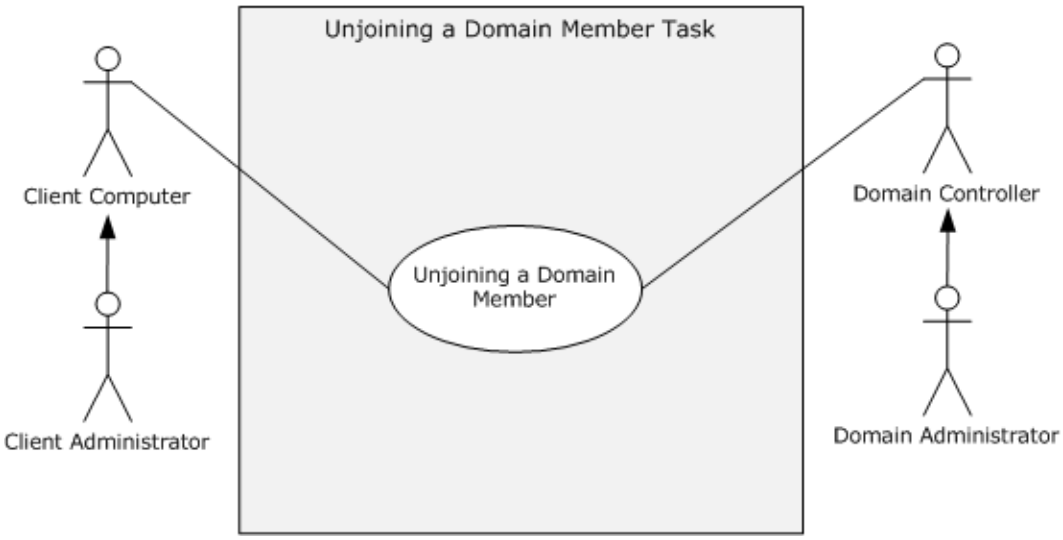

Figure 41: Use case diagram; unjoining a domain member

9.1.3.4 Unjoining a Domain Member - Client Computer

Goal: Unjoin a client computer from a domain.

Context of use: A client administrator wants to unjoin the client computer from the domain it is currently part of, usually to repurpose or decommission the computer.

Primary Actor: The client administrator is the primary actor who initiates unjoining the client computer from a domain. The client administrator invokes commands/applications on the client that initiate the current task.

Direct Actor: The client computer unjoining the domain.

Primary Actor: The client computer unjoining from the domain.

Supporting Actors: All supporting actors are specified in section 9.1.3.2.

Stakeholders and Interests: The stakeholders and interests are specified in section 9.1.3.1.

Preconditions: The client computer MUST have previously completed the domain join task successfully (as described in section 6, 7 or 8) and MUST still be part of the domain.

Minimal Guarantees: If the main success scenario does not successfully finish, the local state of the client computer remains the same as before the unjoin task initialized.

Success Guarantees:

- The state of the computer object on the domain controller (DC) is updated to reflect that the domain client has unjoined from the domain.
- The machine local state of the client computer is updated to reflect that the client has unjoined from the domain.

Triggers: The client administrator initiates this task.

Copyright © 2011 Microsoft Corporation.

Main Success Scenario:

1. The client administrator locates a domain controller (DC).
2. The client computer establishes a SMB/CIFS connection to the domain controller (DC) obtained through the preceding step with the credential of the Domain Administrator (see section 9.4.5.2).
3. The client computer disables the machine account on the domain controller (DC) using the [MS-SAMR] protocol (see section 9.4.5.3).
4. The client computer updates its local state (see section 9.4.5.4).
5. The client computer closes connections (see section 9.4.5.5).
6. The client computer reinitializes local protocols (see section 9.4.5.6).

Extensions: None.

9.2 Task Context

This section describes the relationship between this Task and its environment.

9.2.1 Task Environment

This task assumes that the client is operating in a domain environment. If not, the client cannot be unjoined from the domain.

9.2.2 Task Relationships

This task builds on the Common Task Information (see section 4) which is shared with all of the tasks in this document.

9.2.2.1 Black Box Relationship Diagrams

The following diagram illustrates the system as the client computer unjoins the domain and interacts with the domain controller (DC).

Copyright © 2011 Microsoft Corporation.

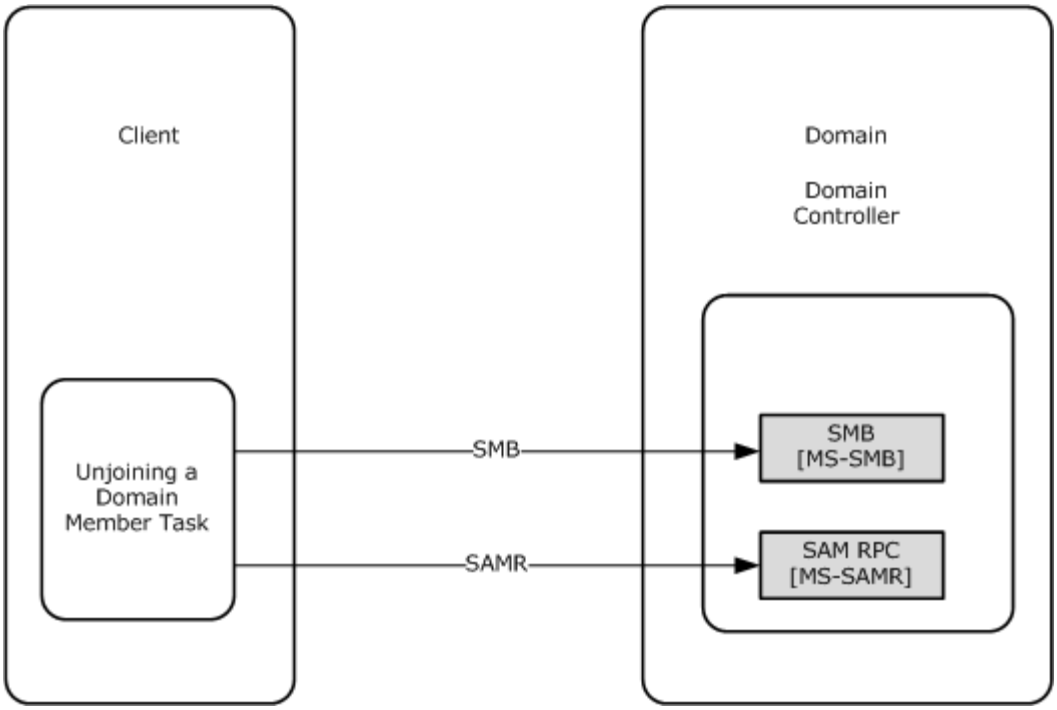

Figure 42: Black box diagram: unjoining a domain member

9.2.2.2 Task Dependencies

Success of this Task also relies on domain controllers being present. The client relies on the local MS-NRPC server to locate a domain controller for the domain. The client relies on the SMB/CIFS protocol to establish a connection to the Domain Controller, and then uses the SAM RPC protocol [MS-SAMR] to update the directory to reflect the departure of the client from the domain.

9.2.2.3 Task Influences

None.

9.2.3 Task Assumptions and Preconditions

This task assumes that the client computer is joined to the domain.

9.2.4 Task Versioning and Capability Negotiation

None.

9.3 Task Architecture

9.3.1 Task Architectural Constraints

This task has the following architectural constraints:

- Only one instance of this task MUST run on the client computer at any given time. The task does not support multiple, parallel executions.

Copyright © 2011 Microsoft Corporation.

- This task MUST NOT run a second time following a *successful* completion as the previous successful completion of the task makes the client computer be no longer domain joined. This violates the precondition for the task to run.
- This task MAY be run a second time following an *unsuccessful* run, as an unsuccessful run makes no changes to the client state.
- This task MUST NOT make assumptions about distributed state, such as the machine account on the domain controller. Notably, if this task is invoked with **TaskInputDisableMachineAccount** (see section 9.3.4) set to TRUE, this task will succeed only if the machine account exists and the processing steps specified in sections 9.4.5.1, 9.4.5.2, and 9.4.5.3 succeed. If the task fails due to a failure in one of these steps, the task MAY be run again with **TaskInputDisableMachineAccount** set to FALSE; the task will only modify local client computer state in that case.

9.3.2 Task Abstract Data Model

This section describes state established, used, and maintained by the processing rules of this Task. State may be volatile or persisted. State may pertain to one, some, or all instances of the Task. The Task's state consists of the values of the named data elements (also called state variables) presented in this section. The overall organization of the data elements, with their names, is the Abstract Data Model. It is intended to facilitate the reader's conceptual understanding of the specification. While a Task's processing rules may depend upon associations established by the structure of its Abstract Data Model, such association can be achieved in other ways. Implementations may depart from this model so long as their external behavior remains consistent with that described in this document.

The following are the in-memory values used by the Task. These values are not persisted.

Name	Type	Description
TaskLocalDomainController	string (Unicode)	Contains the name of the domain controller (DC) the task has determined to use.
TaskLocalSMBSession	SMB/CIFS session	Contains the returned SMB state for the SMB/CIFS session established to the domain controller (DC).

The following client data model variables (section 4.3.1.1) are modified by the task as the final step before it completes a successful run:

- **DomainName.NetBIOS** and **DomainName.FQDN**
- **DomainSid**
- **DomainGUID**
- **SiteName**
- **Password**

9.3.3 Task Abstract Parameters

This section describes data passed to an instance of this task at the time it is invoked or triggered. The parameters consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Parameter. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules might

Copyright © 2011 Microsoft Corporation.

depend upon associations established by the structure of its Abstract Parameters, such association can be achieved in other ways. Implementations can depart from this abstraction so long as their external behavior remains consistent with that described in this document.

The abstract parameters for this task are as follows:

Name	Type	Description	Optional
TaskInputDomainAdministratorName	string (Unicode)	The name of a Domain Administrator that is used to disable the machine account on the domain controller (DC).	No
TaskInputDomainAdministratorPassword	string (Unicode)	The password of the Domain Administrator.	No
TaskInputDisableMachineAccount	boolean	Whether to disable the existing machine account or not.	No

9.3.4 Task Abstract Results

This section describes data returned by an instance of this task to its caller. The results consist of the values of the named data elements presented in this section. The organization of a data element, with its names, is an Abstract Result. It is intended to facilitate the reader's conceptual understanding of the specification. While a task's processing rules might depend upon associations established by the structure of its Abstract Results, such association can be achieved in other ways. Implementations can depart from this abstraction so long as their external behavior remains consistent with that described in this document.

The task returns the following results to the caller:

Name	Type	Description
TaskReturnStatus	integer	This task MUST return 0x00000000 on success. Error statuses generated by a failure during task processing are in the Win32 error space (a long data type), as specified in [MS-ERREF] section 2.2.

Upon a successful task completion, this task also performs the following:

1. Updates the following client variables (section 4.3.1.1):
   - DomainName.NetBIOS
   - DomainName.FQDN
   - DomainGuid
   - DomainSid
   - ClientName
   - Password
   - TrustedDomains

Copyright © 2011 Microsoft Corporation.

The updated values are persisted, as specified in section 4.3.1.1, and are available for other protocols on the client to use.

2. Stops the **NetLogon** Remote Protocol and configures it not to run automatically at every system start up.
3. Notifies the Certificate Autoenrollment system that the machine is unjoining from the domain.
4. Removes the Domain Administrators from the local Administrators group (see invocation of [MS-SAMR] section 3.1.7.1).
5. Notifies the local MS-SNTP protocol that the machine is unjoining from the domain (failures in this step are ignored).
6. Optionally (if *TaskInputDisableMachineAccount* was specified as TRUE), disables the client computer account in the domain.

Upon a failed task completion, this task will attempt to revert any local state changes made (see step 2 in the local state rollback steps, section 9.4.5.4). If the task is unable to disable the computer account upon failure, domain administrator intervention outside the scope of the task may be necessary to disable the account. If the task is unable to revert local state changes, administrator intervention outside the scope of the task may be necessary.

9.3.5 White-Box Relationships

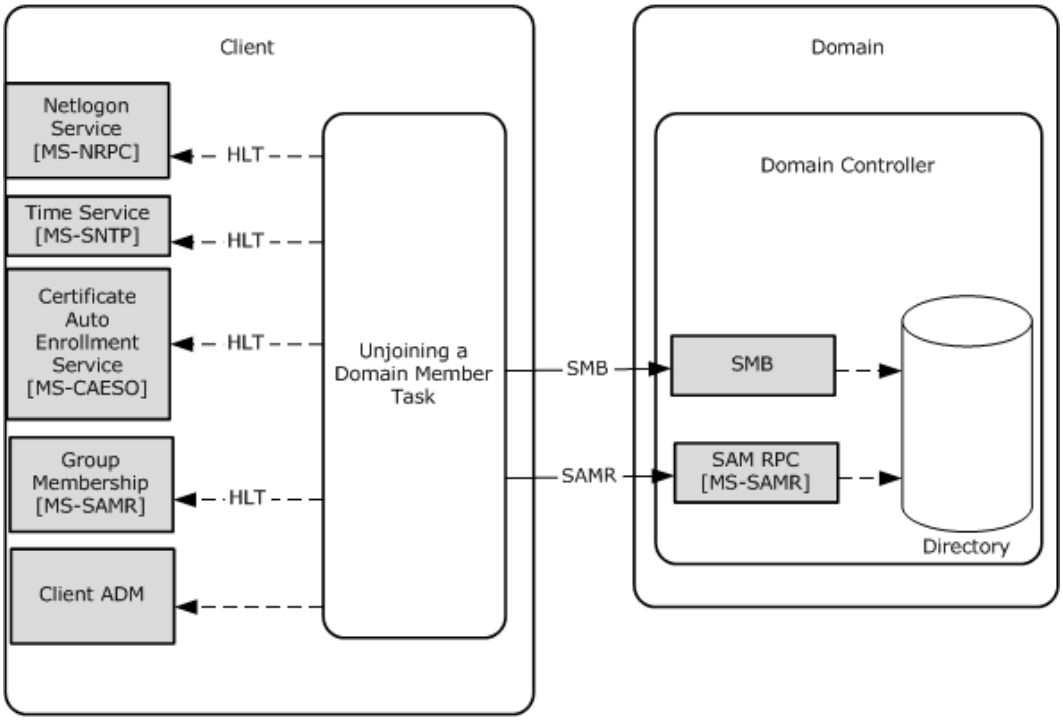

Figure 43: White-box diagram: unjoining a domain member

Copyright © 2011 Microsoft Corporation.

9.3.6 Task Events

9.3.6.1 Task Timers

None.

9.3.6.2 Task Non-Timer Events

None.

9.3.7 Task Architecture and Communication

To unjoin from a domain, a Client Administrator locates a domain controller (DC) using the DsrGetDcNameEx2 method of the local MS-NRPC server and then performs actions against the domain controller (DC). The Client Administrator also updates the machine local state and triggers higher layer events, as shown in the diagram in section 9.3.5.

The following explanations refer to the diagram in section 9.3.5:

Unjoining a Domain Member Task: This box represents the task, including the task ADM and processing. This box interacts with the domain controller for achieving the goals of this task. This box also configures client ADM and client local states.

Client ADM, Group Membership: These boxes are updated by the Unjoining a Domain Member Task box.

Time Service, Cert Auto Enrollment Service, and Netlogon Service: These boxes are signaled by the Unjoining a Domain Task box.

SMB, SAM RPC: These boxes represent the server-side implementations of the protocols used by this task, namely SMB and SAM RPC ([MS-SAMR]).

9.3.8 Task Processing Rules

Task: Unjoining a client computer from a domain.

Abstract Parameters: As specified in section 9.3.3.

Preconditions: The client computer must be joined to the domain.

Main Success Scenario:

1. If **TaskInputDisableMachineAccount** is TRUE:
   - Locate a domain controller (DC) to be used during task processing (see section 9.4.5.1).
   - Establish an SMB/CIFS session to the domain controller (DC) (see section 9.4.5.2).
2. Disable the computer account on the domain controller (DC) (see section 9.4.5.3).
3. Update the local state (see section 9.4.5.4).
4. Close connections (see section 9.4.5.5).

Extensions:

Copyright © 2011 Microsoft Corporation.

1. If the client fails to locate a domain controller, or fails to establish an SMB/CIFS session to the DC, the client MUST terminate the task with an error.

2. If the client fails to disable the computer account, the client MUST terminate the task with an error.

3. If the client fails to update its local state, the client MUST terminate the task with an error.

9.3.9 Task Failure Scenarios

There are three primary failure scenarios:

- **Locating a Domain Controller fails**: If **TaskInputDisableMachineAccount** is set to TRUE, the domain controller (DC) needs to be contacted to disable the computer object. If a DC cannot be located, this will cause this task to fail.
- **Authentication Fails**: If **TaskInputDisableMachineAccount** is set to TRUE, the user needs to be authenticated to disable the computer object. In this situation, the client provides incorrect credentials and is not authenticated, causing the task to fail.
- **Directory Update Fails**: If **TaskInputDisableMachineAccount** is set to TRUE, and updating the **UserAccountControl** attribute on the machine account in Directory fails, the task will fail.
- **Local State Update Fails**: If the client is unable to update its local state, the task will fail.

9.4 Task Details

This section contains the details that complete the descriptions in earlier sections of the document. These are needed to understand and implement this Task.

9.4.1 Task Precondition Details

Not applicable.

9.4.2 Task Initialization of External Entities

None.

9.4.3 Task Event Details

9.4.3.1 Task Timer Details

None.

9.4.3.2 Task Non-Timer Event Details

None.

9.4.4 Task Architectural Details

If the **TaskInputDisableMachineAccount** (section 9.3.3) parameter is passed to unjoin task as TRUE, the Client Administrator will locate a domain controller (DC), and interact with the DC to update the machine object in Directory. The general flow is as follows: The Client Administrator locates a DC using the DsrGetDcNameEx2 method of the local MS-NRPC server. When the DC is determined, the Domain Administrator binds to the DC through the SMB/CIFS protocol using the

Copyright © 2011 Microsoft Corporation.

credential provided to the task (**TaskInputDomainAdministratorName**, **TaskInputDomainAdministratorPassword**, see section 9.3.3). Once assured of valid credentials, the Domain Administrator updates information on the computer object in the domain using SAMR [MS-SAMR].

If the **TaskInputDisableMachineAccount** (section 9.3.3) parameter is passed to the unjoin task as FALSE, the task will proceed without locating a DC or making any modifications other than to local state (section 9.4.5.4).

The following figure shows the network traffic for a typical main success scenario when **TaskInputDisableMachineAccount** has been specified as TRUE. Note that the initial exchange (Locate a DC) is representative only of the traffic between the client and the selected domain controller; additional exchanges that may occur to other domain controllers are not represented. See section 5.4.4 for additional details.

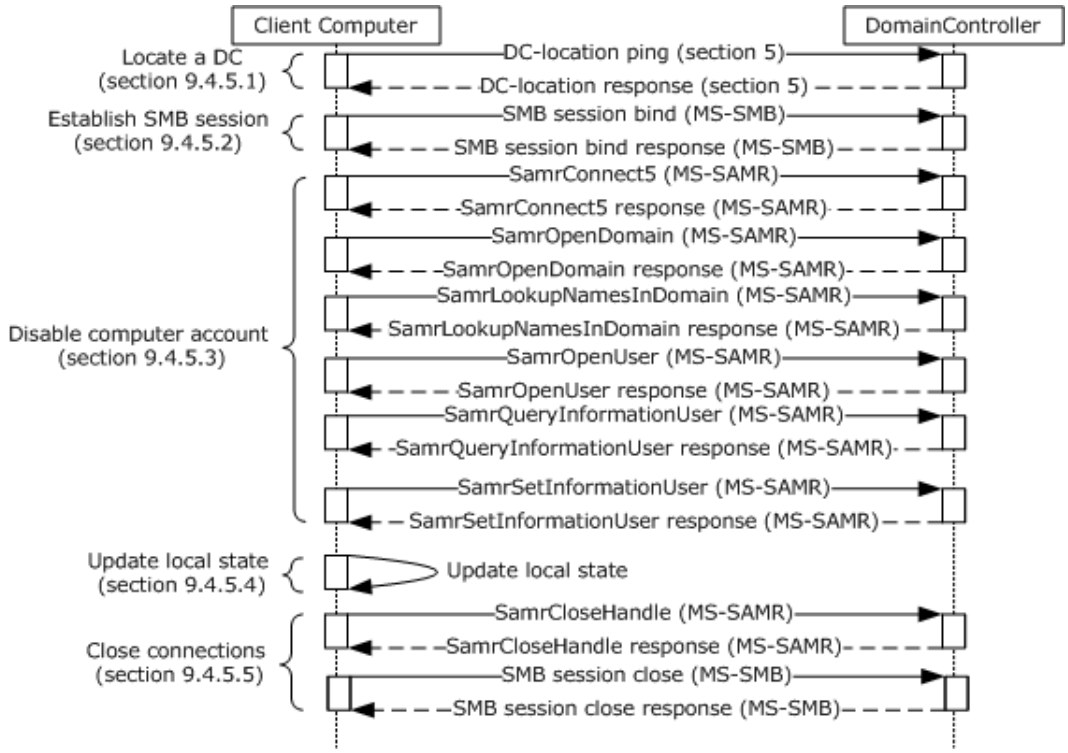

Figure 44: Unjoining client from domain architectural details

The initial state of the client ADM remains that which was persisted as a result of a previously successful domain join task, in addition to which the following are set to NULL: **DomainName**, **DomainSid**, **SiteName**, and **Password**.

9.4.5 Task Processing Rule Details

This section describes details for the steps identified in section 9.3.8. Unless otherwise specified, the processing falls through from one section to the next.

Copyright © 2011 Microsoft Corporation.

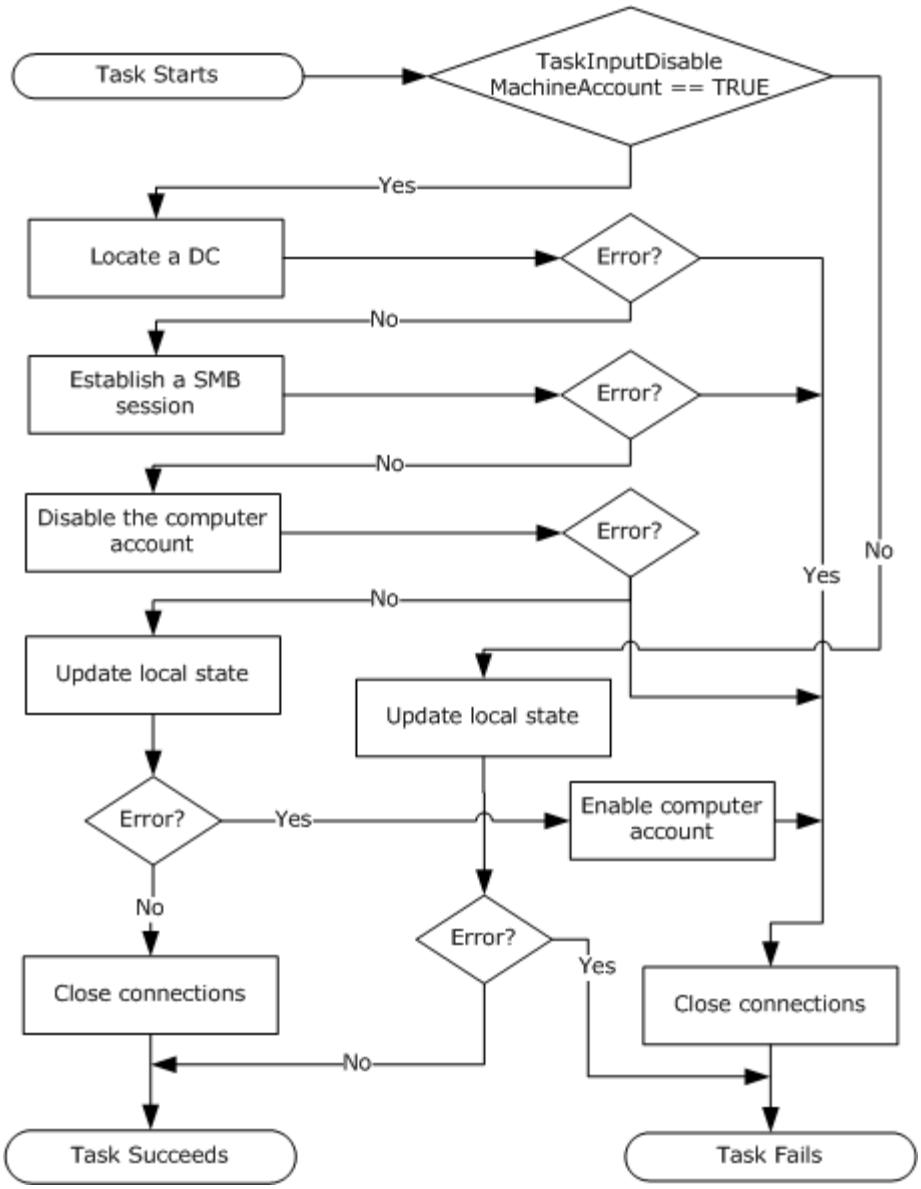

Figure 45: Unjoining client from domain flowchart

9.4.5.1 Locate a Domain Controller

If the **TaskInputDisableMachineAccount** (section 9.3.3) parameter is FALSE, task execution MUST continue at section 9.4.5.4.

If the **TaskInputDisableMachineAccount** (section 9.3.3) parameter is TRUE, the client MUST invoke the **DsrGetDcNameEx2** method on the local MS-NRPC server, specifying the following parameters:

- *ComputerName* = **ComputerName.NetBIOS** (section 4.3.1.1)
- *AccountName* = NULL

Copyright © 2011 Microsoft Corporation.

- *AllowableAccountControlBits* = 0
- *DomainName* = **DomainName.FQDN** (section 4.3.1.1)
- *DomainGuid* = NULL
- *SiteName* = NULL
- *Flags* = (DS_WRITABLE_FLAG | DS_DS_FLAG | DS_LDAP_FLAG | DS_KDC_FLAG) ([MS-ADTS] section 7.3.1.2).

Upon success, the following elements in the task ADM are set as follows:

- **TaskLocalDomainController** = **DomainControllerInfo.DomainControllerName**

If the **DsrGetDcNameEx2** call fails, the current task also fails.

9.4.5.2 Establish SMB Connection

The client MUST establish an authenticated SMB/CIFS connection to the IPC$ share on the **TaskLocalDomainController** domain controller, by invoking [MS-CIFS] section 3.4.4.7 specifying the following parameters:

- **ServerName** = **TaskLocalDomainController.FQDN**
- **UserCredentials** = **TaskInputDomainAdminAccount\TaskInputDomainAdminPassword**

Upon success, the client MUST store the results in **TaskLocalSMBSession**.

9.4.5.3 Disable Computer Account

The following sequence of operation is performed to disable the computer object in the directory:

1. The client MUST bind to the named pipe endpoint `\PIPE\samr`, as presented in [MS-SAMR] section 2.1, on the **TaskLocalDomainController** domain controller.
2. The client MUST connect to the SAM RPC server on the domain controller using one of the **SamrConnect** variants. See [MS-SAMR] section 1.7.2 for information on invoking the **SamrConnect** variants in order to determine version and method supported by the server. See [MS-SAMR] section 3.1.5.1 for using the Open pattern in the SAM interface. The **SamrConnect** method MUST be invoked specifying the following parameters:
   - *ServerName* = **TaskLocalDomainController.FQDN**
   - *DesiredAccess* = GENERIC_ALL
3. The client MUST call **SamrOpenDomain** ([MS-SAMR] section 3.1.5.1.5) specifying the following parameters:
   - *DesiredAccess* = GENERIC_ALL
   - *DomainSid* = **DomainSID** (section 4.3.1.1)
4. The client MUST call the **SamrLookupNamesInDomain** method ([MS-SAMR] section 3.1.5.11.2) to determine the RID for this account, specifying the following parameters:

Copyright © 2011 Microsoft Corporation.

- *DomainHandle* = *DomainHandle* returned in the previous call
- *Count* = 1
- *Names* = **ClientName**
- *Use* = **SidTypeUser**

Upon success, *RelativeIDs* parameter points to the RID of the existing Client account.

5. The client MUST open the account to modify by calling the **SamrOpenUser** method ([MS-SAMR] section 3.1.5.1.9), specifying the following parameters:
   - *DesiredAccess* = GENERIC_ALL
   - *AliasId* = **RelativeIDs** returned in the previous call
6. The client MUST query the current value of **UserAccountControl** on the account using the **SamrQueryInformationUser** method ([MS-SAMR] section 3.1.5.5.6), specifying the following parameters:
   - *UserHandle* = *UserHandle* returned in the previous call
   - *UserInformationClass* = **UserControlInformation** (see [MS-SAMR] section 2.2.7.28)
7. The Client MUST call **SamrSetInformationUser** method ([MS-SAMR] section 3.1.5.6.5) to disable the computer account in the directory, specifying the following parameters:
   - *UserInformationClass* = **UserControlInformation** (see [MS-SAMR] section 2.2.7.28)
   - *Buffer* = **UserInformationClass.UserAccountControl** | USER_ACCOUNT_DISABLED

Upon success, the client has successfully disabled the computer account in the directory.

9.4.5.4 Update Local State

1. Update local state.

   1.a. The client must uninitialize the MS-NRPC protocol [MS-NRPC] and configure it to not run automatically at every boot.

   If this step fails, the task must begin executing step 2 as shown.

   1.b. The client must invoke the Domain Unjoin Processing higher layer triggered event in [MS-SNTP] section 3.1.4.2, which will set the Type abstract data model element to the value "NTP".

   If this step fails, the task must begin executing step 2 as shown.

   1.c. The Client must invoke the "Certificate Autoenrollment Task" task ([MS-CAESO] section 4) specifying the *IsUnjoiningDomain* input parameter ([MS-CAESO] section 4.3.3) to be TRUE.

   If this step fails, the task must begin executing step 2 as shown.

   1.d. The client MUST invoke the Domain unjoin processing event in MS-SAMR ([MS-SAMR] section 3.1.7.2), specifying the following parameter:

Copyright © 2011 Microsoft Corporation.

DomainSID = **DomainSID**

If this step fails, the task must begin executing step 2 as shown.

1.e. The client MUST update the following ADM values (section 4.3.1.1) to the values specified as follows. These values MUST be modified in a single atomic update.

- **ClientName** = NULL
- **DomainName.FQDN** = NULL
- **DomainName.NetBIOS** = "Workgroup"
- **DomainSid** = NULL
- **ForestNameFQDN** = NULL
- **DomainGuid** = NULL
- **SiteName** = NULL

2. Rollback in case of error.

The following steps are executed only upon a failure of one of the substeps in section 1 as shown, in order to reverse those state changes that were previously made during step 1 processing.

2.a. The client MUST restore the client ADM variables updated in step 1d as shown to their original values.

2.b. The client MUST initialize the MS-NRPC protocol, and configure it to run at every system restart.

2.c. The client must invoke the Domain Join Processing higher layer triggered event in [MS-SNTP] section 3.1.4.1, which will set the **Type** abstract data model element to the value "NT5DS".

2.d. The client must invoke the "Certificate Autoenrollment Task" task ([MS-CAESO] section 4) specifying the IsUnjoiningDomain input parameter ([MS-CAESO] section 4.3.3) to be FALSE.

2.e. The client MUST invoke the domain join processing event in [MS-SAMR] section 3.1.7.1 with the parameters set as follows:

DomainSID = **DomainSID**

9.4.5.5 Close Connections

The client MUST close any [MS-SAMR] handles opened earlier in section 9.4.5.3 by calling **SamrCloseHandle** method ([MS-SAMR] section 3.1.5.13.1) and passing the handle as the parameter. The client MUST close all open handles irrespective of previous errors in this call sequence.

If an SMB/CIFS session was previously established (see section 9.4.5.2), the client MUST disconnect as described in [MS-CIFS] section 3.4.4.8, specifying **TaskLocalSMBSession**.

9.4.5.6 Reinitialize Local Protocols

The client MUST reinitialize local protocols using the steps specified in section 6.4.5.8.

Copyright © 2011 Microsoft Corporation.

9.5 Task Security

If this task is invoked with **TaskInputDisableMachineAccount** set to FALSE, this MAY result in the client computer machine account being left in an undesirable state (not disabled); administrative action may be necessary to make the appropriate changes to the machine account outside the scope of the task.

For more information, refer to the Security section of this specification and the Security sections of the referenced protocol Technical Documents (TDs).

Copyright © 2011 Microsoft Corporation.

10 Security

This section documents security issues common to all tasks that are not otherwise described in the Technical Documents (TDs) for the protocols used in the task. It does not duplicate what is already in the protocol TDs unless there is some unique aspect that applies to the system as a whole.

Interacting with the domain controller brings with it certain constraints that affect the security of the distributed system and domain as a whole. Since the domain controller serves as the root of trust for the entire domain, and potentially additional domains based on trust relationships, great attention has to be paid to the correctness of the implementation of all the member protocols. A failure in the implementation that allows an exploit could compromise the entire domain.

When beginning any interaction with the domain controller, the domain client is in a tenuous state. It has to establish a connection and authenticate in a way that prevents attackers from manipulating the messages exchanged between the client and the domain controller. For all protocols involved in these situations consider the security conditions described in the following section.

10.1 Untrusted Data

Some protocols used to start the domain interactions are by necessity untrusted--for example, responses to DNS queries are typically not secure. It is important to limit the use of these untrusted data to cases where malicious tampering can be mitigated. For example, a malicious DNS reply could come back to a client, sending it to a malicious, fake domain controller. This is mitigated by the client requiring strong, mutual authentication with the domain controller.

The influence of untrusted data can be more subtle than an attack on name resolution. When exchanging data (for example, a query from the directory) and the communication channel is not encrypted, perform an integrity check. An attacker might modify data in flight and cause unexpected behavior. Since it is rarely possible to determine only what is security-relevant data, treat all data as vital. A server might be making an authorization decision based on an LDAP query from the directory; if an attacker was able to change that, the decision would be subverted.

10.2 Authentication

It is as important to authenticate a client to a server as it is to authenticate the server to the client. Legacy authentication schemes, and even some more modern ones, are focused on one party or the other. While it is clearly in the server's interest not to disclose inappropriate information to the wrong client, it is equally as risky for a client to connect to a malicious server.

Mutual authentication is possible only when the client can express what server it is trying to contact. Domain clients can only be assured of mutual authentication when they use well-formed service principal names to specify the server to which they are connecting.

Copyright © 2011 Microsoft Corporation.

11 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft Windows NT® 3.1 operating system
- Microsoft Windows NT® 3.5 operating system
- Microsoft Windows NT® 3.51 operating system
- Microsoft Windows NT® 4.0 operating system
- Microsoft Windows® 2000 operating system
- Windows® XP operating system
- Windows Server® 2003 operating system
- Windows Server® 2003 R2 operating system
- Windows Vista® operating system
- Windows Server® 2008 operating system
- Windows® 7 operating system
- Windows Server® 2008 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 3.1.4.4: Windows products from Windows NT 4.0 to Windows Server 2008 all add the well-known group "Domain Admins" to the administrators group on the domain client. However, this is specific to the Windows implementation and not a requirement for domain membership in general.

<2> Section 3.3: For Windows, the issuing authority is the domain.

<3> Section 4.3.1.1: All Windows DCs implement site awareness.

<4> Section 5.4.4.1: Windows 2000, Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 include the machine account name of the domain client in the LDAP "Ping". Windows NT 4.0 does not support LDAP "Ping".

<5> Section 5.4.4.1: Windows 2000 through Windows Server 2008 domain clients retry once per SRV entry returned. Windows NT 4.0 domain clients do not perform DNS-based domain controller location.

Copyright © 2011 Microsoft Corporation.

<6> Section 5.4.4.1: Windows 2000 through Windows Server 2008 domain clients cache negative results for no more than five minutes. Windows NT 4.0 domain clients do not perform DNS-based domain controller location.

<7> Section 5.4.5.3: Windows 2000, Windows XP, Windows Server 2003, Windows Server 2003 R2, and Windows Vista clients do not include the NETLOGON_NT_VERSION_WITH_CLOSEST_SITE bit in the LDAP/Mailslot ping message.

<8> Section 5.4.5.3: Windows 2000 and later domain clients retry once per SRV entry returned. Windows NT 4.0 domain clients do not perform DNS-based domain controller location.

<9> Section 5.4.5.6: If the FQDN(1) and the NetBIOS name are both available, the name format to be returned is chosen to match the format of the domain name that was specified in the abstract parameter **TaskInputDomainName**. If only one of the names is available, that name is used.

<10> Section 6.4.5.5: All Windows client invocations follow these steps. The versions of which Windows server products that support these functions can be found in [MS-NRPC].

<11> Section 6.4.5.8: All versions of Windows accomplish re-initialization of the [MS-SNTP] protocol (on either domain join or unjoin) by rebooting the client machine.

<12> Section 7.4.5.4: Windows products from Windows 2000 Server to Windows Server 2008 R2 all generate an ASCII string of randomly chosen characters. For more information, see [RFC4086]. Each character's ASCII code is between 32 and 122 inclusive. When randomly generating a password string, the server generates 120 characters. Each character is generated using the algorithm specified in [FIPS186-2] appendix 3.1.

[FIPS186-2] appendix 3.1 describes a pseudo-random number generator (PRNG) that can use either DES or SHA-1. Windows uses a SHA-1-based PRNG to satisfy FIPS 140-2 level 2 cryptographic module certification requirements.

In the PRNG description of appendix 3.1, G is constructed from SHA-1 with the first parameter as the initial value for the SHA-1 registers, and the second parameter is the data input to be hashed.

Integer b is replaced with 160.

XKEY is determined by a call to an RC4-based PRNG.

The variable q is not used in the general purpose version of [FIPS186-2] (see appendix 6.9 General Purpose Random Number Generation).

XSEEDj is also determined by a call to an RC4-based PRNG for every block output by the [FIPS186-2] PRNG.

The variable *m* is the number of blocks that can be output by the [FIPS186-2] PRNG before a non-NULL value is passed to XSEEDj. The Windows implementation sets it to the shortest possible value, which is 1.

<13> Section 7.4.5.5: Windows 2000, Windows XP, Windows Server 2003, Windows Server 2003 R2, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 update domain information in this way. Windows NT 3.1, Windows NT 3.5, Windows NT 3.51, and Windows NT 4.0 did not.

<14> Section 7.4.5.5: Windows domain clients beginning with Windows 2000 use GSS-SPNEGO with Kerberos. Windows NT 4.0 domain clients do not perform this action at all. Other domain clients are known to use LDAP with GSSAPI (Kerberos) only, so that any domain controller implementation should be prepared for any valid form of authentication.

Copyright © 2011 Microsoft Corporation.

<15> Section 7.4.5.5: Windows domain clients beginning with Windows 2000 set these values. Windows NT 4.0 domain clients set none of these. Windows 2000 domain clients do not specify the RestrictedKrbHost form of SPN.

<16> Section 8.4.5.4: Windows products from Windows 2000 Server to Windows Server 2008 R2 all generate an ASCII string of randomly chosen characters. For more information, see [RFC4086]. Each character's ASCII code is between 32 and 122 inclusive. When randomly generating a password string, the server generates 120 characters. Each character is generated using the algorithm specified in [FIPS186-2] appendix 3.1.

[FIPS186-2] appendix 3.1 describes a pseudo-random number generator (PRNG) that can use either DES or SHA-1. Windows uses a SHA-1-based PRNG to satisfy FIPS 140-2 level 2 cryptographic module certification requirements.

In the PRNG description of appendix 3.1, G is constructed from SHA-1 with the first parameter as the initial value for the SHA-1 registers, and the second parameter is the data input to be hashed.

Integer b is replaced with 160.

XKEY is determined by a call to an RC4-based PRNG.

The variable q is not used in the general purpose version of [FIPS186-2] (see appendix 6.9 General Purpose Random Number Generation).

XSEEDj is also determined by a call to an RC4-based PRNG for every block output by the [FIPS186-2] PRNG.

The variable m is the number of blocks that can be output by the [FIPS186-2] PRNG before a non-NULL value is passed to XSEEDj. The Windows implementation sets it to the shortest possible value, which is 1.

Copyright © 2011 Microsoft Corporation.

12 Change Tracking

This section identifies changes that were made to the [MS-DISO] protocol document between the May 2011 and June 2011 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

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

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

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

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

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

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

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

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

Copyright © 2011 Microsoft Corporation.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

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

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.2 References	Added explanatory statement regarding the removal of the publishing year from Microsoft Open Specification document references.	N	Content updated.

Copyright © 2011 Microsoft Corporation.

13 Index

Copyright © 2011 Microsoft Corporation.

Copyright © 2011 Microsoft Corporation.

L

N

O

P

R

Copyright © 2011 Microsoft Corporation.

S

T

U

Copyright © 2011 Microsoft Corporation.

V

W

Copyright © 2011 Microsoft Corporation.