The Library for System Solutions Directory, Naming and Time

Document Number GG24-4104-00

August 1994

International Technical Support Organization
Poughkeepsie Center

Take Note!

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xiii.

First Edition (August 1994)

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. H52 Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

Abstract

This document is part of The Library for Systems Solutions, which is intended for technical professionals involved in defining solutions to data processing problems in multiple configuration environments, with multiple software platforms, including heterogeneous distributed environments.

Several hardware, software, and architecture technologies are analyzed to establish a knowledge repository for defining solutions to problems involving directory, naming and time. The most common software and hardware platforms are described in tabular form with those solutions that are available today.

(150 pages)

Contents

Abstract . . . iii

Special Notices . . . xiii

Preface . . . xv
How This Document is Organized . . . xv
Related Publications . . . xv
International Technical Support Organization Publications . . . xvi
Acknowledgments . . . xvi

Part 1. Technology . . . 1

Chapter 1. Introduction . . . 3
1.1 Are the Entities Global or Local? . . . 3
1.2 Directories . . . 4
1.3 Naming . . . 4
1.4 Time . . . 5
1.5 Technology Decisions . . . 5
1.6 Open Distributed System Structure . . . 5
1.7 Client/Server Considerations . . . 6
1.7.1 Network Transparency . . . 7

Chapter 2. Directory Technologies and Standards . . . 9
2.1 Directory Requirements . . . 10
2.2 The Internet Directory . . . 10
2.2.1 What is the Internet? . . . 10
2.2.2 Importance of Internet . . . 12
2.2.3 Internet Naming . . . 12
2.2.4 Relationship of the Internet to OSI . . . 14
2.2.5 Internet Trends and Directions . . . 15
2.3 The X.500/ISO 9594 International Standard . . . 15
2.3.1 Directory Overview Model . . . 16
2.3.2 Directory Information model . . . 17
2.3.3 Schema, Objects, Rules . . . 22
2.3.4 Directory Schema . . . 26
2.3.5 Directory Service . . . 27
2.3.6 Distributed Operations . . . 31
2.3.7 Service Controls . . . 36
2.3.8 Search Guide . . . 37
2.3.9 Security . . . 38
2.3.10 Areas Excluded from the 1988 Directory Standard . . . 38
2.3.11 NADF, GOSIP, EWOS, and Other Interest Groups . . . 40
2.4 APPN Overview . . . 41
2.4.1 LEN and APPN . . . 41
2.4.2 Names . . . 46
2.4.3 Addresses . . . 46
2.4.4 Domains . . . 48
2.4.5 Node Types . . . 48
2.5 The IBM Open Blueprint . . . 54
2.6 OSF/DCE Directory . . . 56
2.6.1 Cell . . . 57

2.6.2 Directory Services and the Cell . . . 58
2.6.3 Naming Environment . . . 59
2.6.4 Cell Directory Service (CDS) . . . 63

Chapter 3. Naming Technologies and Standards . . . 73
3.1 Global Naming Standards . . . 73
3.2 Naming in the IBM Open Blueprint . . . 74
3.2.1 Universal Naming . . . 74
3.2.2 Concatenated Naming . . . 75
3.2.3 Contextual Names . . . 76
3.3 ISO Naming . . . 76
3.3.1 ISO 7498 . . . 77
3.3.2 ISO 8824 . . . 78
3.3.3 ISO 9834 . . . 80
3.3.4 ISO 8879 and ISO 9070 . . . 82
3.3.5 Ancillary ISO Standards . . . 83
3.4 ASN.1 Object Identifiers . . . 83
3.4.1 Role of ASN.1 Object Identifier . . . 84
3.4.2 Examples . . . 84
3.4.3 Subsequent Arcs – CCITT . . . 86
3.4.4 Subsequent Arcs – ISO . . . 86
3.4.5 Subsequent Arcs – Joint ISO and CCITT . . . 87
3.4.6 Use of ASN.1 OIDs . . . 87
3.4.7 Example . . . 87
3.4.8 IBM Use of OIDs . . . 89
3.4.9 Subtrees in IBM . . . 89
3.4.10 Encoding an ASN.1 OID . . . 90
3.5 OSF Naming, CORBA Naming, and Naming APIs . . . 91
3.5.1 OSF Naming . . . 91
3.5.2 CORBA Naming . . . 92
3.5.3 Naming APIs . . . 93

Chapter 4. Time Services Technologies . . . 95
4.1 Time in IBM′s Open Blueprint . . . 95
4.2 MVS Time and Time Services . . . 96
4.3 DCE Distributed Time Services (DTS) . . . 96
4.3.1 Why a Time Service . . . 97
4.3.2 How Does It Work ? . . . 98

Part 2. Solutions . . . 115

Chapter 5. Directory, Naming and Time Solutions . . . 117
5.1 Directory and Naming Implementations . . . 117
5.2 Time Service Implementations . . . 120

Part 3. Appendixes . . . 121

Appendix A. OSI Reference Model . . . 123

Appendix B. Time in MVS . . . 127
B.1 Time and Date Concepts . . . 127
B.1.1 Why Time and Date Corrections Are Necessary . . . 128
B.1.2 Atomic Time . . . 128

B.1.3 Universal Time Coordinated (UTC) 128
B.2 Sysplex Timer, TOD clock, and MVS/ESA Clock Concepts 129
B.3 MVS/ESA Time 130
B.3.1 Time Representations and Guidelines 131
B.3.2 Time Compatibility between MVS Releases 131
B.4 Time Adjustments 132
B.4.1 Adjustments to the TOD Clock 132
B.4.2 Adjustments From an External Time Source 132
B.4.3 Planned Adjustments to Time Offsets 133

Appendix C. Overview of ASN.1 135
C.1.1 Background 135
C.1.2 Purpose 135
C.1.3 Selected Terminology 135
C.2 ASN.1 Structural Elements 135
C.2.1 ASN.1 Character Set 136
C.2.2 Reserved Words 136
C.2.3 Productions 137
C.2.4 Example of ASN.1 Production 137
C.2.5 Miscellaneous Rules 137
C.2.6 ASN.1 Data Types 137
C.2.7 Modules 141
C.2.8 Macros 141

Appendix D. Overview of ISO 6523 143

Appendix E. Overview of X.400 145

Index 147

Figures

1. The Open Blueprint . . . 6
2. Client/Server Architecture . . . 7
3. Directory Entry and Objects . . . 18
4. Directory Information Tree (DIT) . . . 19
5. Alias Entries and Alias Names . . . 20
6. Directory Entries and Attributes . . . 20
7. Directory Entry Structure . . . 21
8. Relative Distinguished Names . . . 22
9. Selected Attribute Types - 1 . . . 23
10. Selected Attribute Types - 2 . . . 24
11. Selected Object Classes . . . 25
12. Subclasses of Object Classes . . . 25
13. DIT Structure Rules . . . 26
14. Directory Schema . . . 26
15. Directory Service Operations Overview . . . 27
16. Abandon . . . 27
17. Read . . . 28
18. Compare . . . 28
19. List . . . 29
20. Search . . . 29
21. Result from Search . . . 30
22. Add Entry . . . 30
23. Remove Entry . . . 30
24. Modify Entry . . . 31
25. Modify RDN . . . 31
26. Serial Chaining . . . 32
27. Parallel Chaining or Multicasting . . . 32
28. Referrals Pursued by DUA . . . 33
29. Referrals Pursued by DSA . . . 33
30. Knowledge References . . . 34
31. Superior Reference . . . 35
32. Subordinate Reference . . . 35
33. Cross References. . . . 36
34. Two PS/2s Forming a LEN Connection . . . 41
35. The Basic LEN Connection . . . 42
36. LEN End Nodes Connected to an Intermediate Node . . . 42
37. VTAM/NCP Providing the Intermediate Routing Function for LEN End Nodes . . . 43
38. Advanced Peer-to-Peer Networking with Three Nodes . . . 43
39. APPN Network with Different Node Types . . . 44
40. Advanced Peer-to-Peer Networking . . . 45
41. Composite Network Node with APPN Network Node Appearance . . . 45
42. Session with Several Session Stages . . . 47
43. Composite Network Node Acting as an Interchange Node . . . 52
44. HPR Nodes and HPR Subnets . . . 54
45. OSF DCE Directory Service . . . 56
46. Multi-Cell Environment . . . 57
47. Components of the Directory Service . . . 58
48. Interaction of CDS and GDA . . . 59
49. Representation of a User . . . 59
50. Comparison of Name Representation . . . 61

51. Example of a Cell Namespaces in a DCE Namespace 61
52. Directories Created for Each Cell 62
53. Components of a Cell Directory Service 63
54. Sample CDS Lookup 64
55. Components of a CDS Server Node 66
56. Logical and Physical View of a Namespace 67
57. Clearinghouse Object Entries and Clearinghouses 68
58. Soft Link and its Resolution 69
59. Child Pointers and Directories 70
60. Universal Naming Examples 75
61. Concatenated Naming Example 76
62. Distributed Time Services as a DCE Component 97
63. Client/Server Time Synchronization 98
64. Time Intervals 99
65. Faulty Server 100
66. Time Servers 101
67. Courier Server 102
68. DTS Structure 103
69. Absolute Time 105
70. Variation of the ISO Time Format 106
71. Relative Time Structure 107
72. Period of Time 108
73. DTS Daemon and TP Process RPC Calling Sequence 112

Tables

1. Correspondence between Nodes and Attributes 82
2. Values for Component C 85
3. CCITT Arc: Values for Component C+1 85
4. ISO Arc: Values for Component C+1 85
5. Analysis of Proposed OID 87
6. IBM OID Subtrees for 1 3 18 Path 89
7. IBM Internal Subtrees 89
8. Directory and Naming Technologies supported by Various Platforms . 117
9. Summary of Novell Netware Directory Services (Source: LAN Systems Inc.) 118
10. Summary of Banyan Streettalk (Source: LAN Systems Inc.) 118
11. Time Service Technologies supported by Various Platforms 120
12. Timestamps in MVS 131

Special Notices

This publication is part of The Library for Systems Solutions, which is intended for technical professionals involved in defining solutions to data processing problems in multiple configuration environments, and multiple software platforms. In particular, this publication contains information about both the traditional and more recent developments in the fields of directory, naming and time, and products that use these technologies.

Whenever a product is mentioned, the information about it is not intended as specification of any programming interfaces that are provided by that product. See the PUBLICATIONS section of the IBM Programming Announcement for each named IBM product for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 208 Harbor Drive, Stamford, CT 06904 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

The following terms, which are denoted by an asterisk (*) in this publication, are trademarks of the International Business Machines Corporation in the United States and/or other countries:

Advanced Peer-to-Peer Networking	APPN
ES/3090	ES/9000
IBM	MVS/ESA
OS/2	OS/400
PS/2	S/390
Sysplex Timer	System/390

The following terms, which are denoted by a double asterisk (**) in this publication, are trademarks of other companies:

Andrew File System, Encina	Transarc Corporation
Apollo, Hewlett-Packard, HP-UX, MPE/iX, HP	Hewlett-Packard Company
AViiON	Data General
Banyan, Vines, Streettalk	Banyan Systems, Inc
DAZEL, DACM	Atrium Technologies
DECdns, DECnet, VAX, Alpha	Digital Equipment Corporation
DIR-X	Siemens Nixdorf Informationssysteme AG
Ellery Open Systems	Ellery Systems
Hitachi	Hitachi Ltd.
Macintosh	Apple Computer, Inc
Microsoft, Windows	Microsoft Corporation
NCR	NCR Corporation
NEC	NEC Corporation
Novell, IPX, Netware	Novell Corporation
Olivetti	Ing C. Olivetti & C., S.p.a.
Oracle	Oracle Corporation
OSF, OSF/1, Motif	Open Software Foundation
SCO	The Santa Cruz Operation, Inc.
Stratus	Stratus (complete name ??)
SunOS, SPARCstation, Network File System, NFS	Sun Microsystems, Inc.
SYBASE	Sybase, Inc.
Tandem	Tandem

Other trademarks are trademarks of their respective companies.

Preface

This document is part of The Library for Systems Solutions. It contains information about both the traditional and more recent developments in directory, naming, and time technologies and products that use these technologies. In addition, it contains tabulated information about software and hardware elements that are part of data processing solutions, and the corresponding directory, naming, and time solutions.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “Introduction”

  This provides an overview of current directory, name, and time technologies.
- Chapter 2, “Directory Technologies and Standards”

  This provides an overview of directory technologies.
- Chapter 3, “Naming Technologies and Standards”

  This provides an overview of naming technologies.
- Chapter 4, “Time Services Technologies”

  This provides an overview of timing technologies.
- Chapter 5, “Directory, Naming and Time Solutions”

  This chapter describes the directory, naming, and time solutions available.
- Appendix A, “OSI Reference Model”

  This appendix describes the OSI base reference model.
- Appendix B, “Time in MVS”

  This appendix describes the timekeeping in MVS/ESA.
- Appendix C, “Overview of ASN.1”

  This appendix describes the Abstract Syntax Notation One (ASN.1).
- Appendix D, “Overview of ISO 6523”

  This appendix describes the ISO 6523 global naming standard.
- Appendix E, “Overview of X.400”

  This appendix describes the X.400 electronic mail naming standard.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- The Library for System Solutions:

  Computing Technology Reference, GG24-4100 [1])
  Application Development Reference, GG24-4101
  Workload Management Reference, GG24-4102
  Data Reference, GG24-4103
  Printing and Viewing Reference, GG24-4105

Security Reference, GG24-4106
End User Interface Reference, GG24-4107
Multimedia Reference, GG24-4108 [1])
Image Processing Reference, GG24-4109
Networking Reference, GG24-4110
LAN Reference, GG24-4111
Office Reference, GG24-4112
Systems Management Reference, GG24-4113

[1]) Available at a later date.

- *Planning for the 9037 Sysplex Timer*, GA23-0365

International Technical Support Organization Publications

- *APPN Architecture and Product Implementations Tutorial*, GG24-3669

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

Bibliography of International Technical Support Organization Technical Bulletins, GG24-3070.

To get listings of redbooks online, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order Redbooks

IBM employees may order redbooks and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their IBM branch office.

You may order individual books, CD-ROM collections, or customized sets, called GBOFs, which relate to specific functions of interest to you.

Acknowledgments

The advisor for this project was:

Erik P. Olsen
International Technical Support Organization, Poughkeepsie Center

The author of this document was:

Robert Wilcox IBM Germany

This publication is the result of a residency conducted at the International Technical Support Organization, Poughkeepsie Center.

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Arnie W. Powell IBM Open Distributed Marketing, New York
Diane Baron IBM Open Distribution Systems Strategy, New York
R. L. Sikkema IBM Distributed Systems Development, Texas

Part 1. Technology

Chapter 1. Introduction

Directory, naming, and time refer to services and resources that are absolutely essential for every data processing system and are present in any such system.

Even the smallest computing system contains many entities that must be addressed and accessed. Files and terminals are examples of such entities. In order to address these, it is necessary to identify them in some way. Because many of these entities must be accessed and recognized by the human operators of a computing system, it often makes sense to identify entities with designations that make sense to humans. This is not absolutely necessary, of course, and in fact many object identifiers are optimized for machine rather than human readability.

Once objects have been named, there must be some sort of roadmap for finding the objects, given the name. That roadmap is a directory. Directories take many forms, but the two most common ones are the “white pages,” where one knows the name of entity one wants to access, and “yellow pages,” where one knows the category to which an entity belongs, but not necessarily the name of the entity itself.

Finally, even though many remember when it was possible to boot up a PC without setting its clock, it is clearly essential to know when certain events – such as a file update – have taken place. Even if the clock is not synchronized with a standard time, such as Universal Time Coordinated (UTC, see 4.3.2.4, “Universal Time Coordinated (UTC)” on page 100), it is still very useful to know the sequence in which events have happened. This is accomplished with the aid of time services.

Generally, the technology and architecture of directory, naming, and time services are determined by the hardware and software platforms used, and leave very little opportunity for choice by the user. Compatibility between different platforms is usually limited or non-existent. These caveats do not necessarily apply to open systems architectures, which offer considerable scope for interoperability.

This document is divided into two sections. The first section will discuss the various technologies available for directory, naming, and time, while the second section one will discuss the currently available implementations.

The purpose of this chapter is to give an overview of the concepts that are to be described in the rest of the document. The discussions here are made in terms of familiar concepts and are not necessarily formally correct or complete. This is done to set the expectation for what is to appear later in the document.

1.1 Are the Entities Global or Local?

It can be useful to think of these three subjects as having two possible qualities, local or global. While not totally rigorous, local can be defined as belonging to a single system or group of similar coupled systems that are compatible, while global can be thought of as applying to an undefined number of systems whose characteristics are not necessarily compatible.

Obviously, a system that requires a local solution is much easier to implement than one requiring a global one. Also, it is not necessarily a problem if a local system uses a proprietary architecture, since it does not interface directly with incompatible systems.

A global solution is much more difficult to find, since it must include support for many incompatible subsystems. It is therefore not surprising that global solutions tend to be based on open, standardized architectures and technologies. This document is weighted heavily toward the discussion of global architectures and open systems simply because they produce the greater problems and demand the more difficult choices.

As a shorthand way of looking at this, we can assume homogeneous systems and homogeneous networks can tolerate local solutions, while heterogeneous distributed systems will probably demand global types of directory, naming, and time. This does not preclude a homogeneous system from adopting open architectures of course.

1.2 Directories

Directories can be thought of as lists of names that provide additional information about each of the names in the list. A simple directory, such as a DOS directory, provides information about the physical location of a file, for example.

A more complex directory could be one that provides information about any type of resource anywhere in a distributed, heterogeneous computing network. This could be thought of as the central roadmap for navigating one′s way around the complex web of electronic mail service, network environments and resources that can be located anywhere in the network. Obviously, such a directory is much more complex and powerful than a simple MSDOS file directory.

Whereas a primitive OS/360 system could survive without a catalog, it is inconceivable that it would be possible to run a distributed system without some kind of directory.

1.3 Naming

Naming is the process of giving objects unique identifiers. Traditionally, we have used names to identify files, volumes, and network entities, such as terminals. However, with the advent of distributed systems and LANs, names have been assigned to more and more objects. Ethernet controllers and token ring cards are being given unique names that can be read and managed electronically, for example. It is now even likely that the microchips themselves will be given electronically readable names. This occurs not as a result of a requirement for better systems or network management, but for the more banal reason of discouraging their theft.

It is in naming that the global/local dichotomy takes on considerable significance. The scope of a name can vary enormously; it can be very narrow or so wide as to encompass the universe, or at least that part relevant to human activity. For example, there are probably millions of files in the world called C:CONFIG.SYS. Such a file name is only unique for a very restricted domain, i.e., one that encompasses only the root directory of a C: disk of a single PC. At

the other extreme, there is probably only one Internet user name joe_bloggs@podunk.univ.edu in the entire world. That is because the scope of an Internet name is very wide, encompassing names of users on many hundreds of thousands of host computers.

Because directories are essentially just lists of names supplemented by additional information, it is difficult to totally separate discussions of the two subjects. There are some naming standards that can be described by themselves while others, such as the Distinguished Naming Standard (DNS), are inextricably linked with the directories that support them. For that reason, DNS is discussed in Chapter 2, "Directory Technologies and Standards."

1.4 Time

Time is also a very important concept in a data processing system of any sort. It is often essential to know the sequence in which events happened in a system. This can only be accomplished if there is some form of consistent timing present in the system. It is essential that a distributed system have consistent and accurate timing and have appropriate services for obtaining and processing time information.

1.5 Technology Decisions

Ideally, one would like to have directory, naming, and time services that are common across all platforms, thus allowing transparent access to resources anywhere in a distributed, heterogeneous network. At the time this document was written (mid 1994), this was still unrealized, although many vendors were engaged in working towards such integration, either individually or working together in consortia. Many of the decisions will be forced by the need to accommodate technologies already in use, possibly by providing synchronizing services to be able to treat several separate directory types as a single one, for example.

1.6 Open Distributed System Structure

IBM*'s software strategy for open client/server or distributed computing is described in the *Open Blueprint*. It is represented in Figure 1 on page 6. New areas of distributed computing technology continue to be explored, and the structure itself will evolve to include those new technologies. Therefore, the figure will change over time, but those changes are likely to add or rearrange elements, rather than change any of the fundamental concepts or services described in this book.

This figure may seem familiar because it builds on the *Networking Blueprint* announced in March 1992. The figure differs slightly from that announcement, because it had a more specific focus on networking.

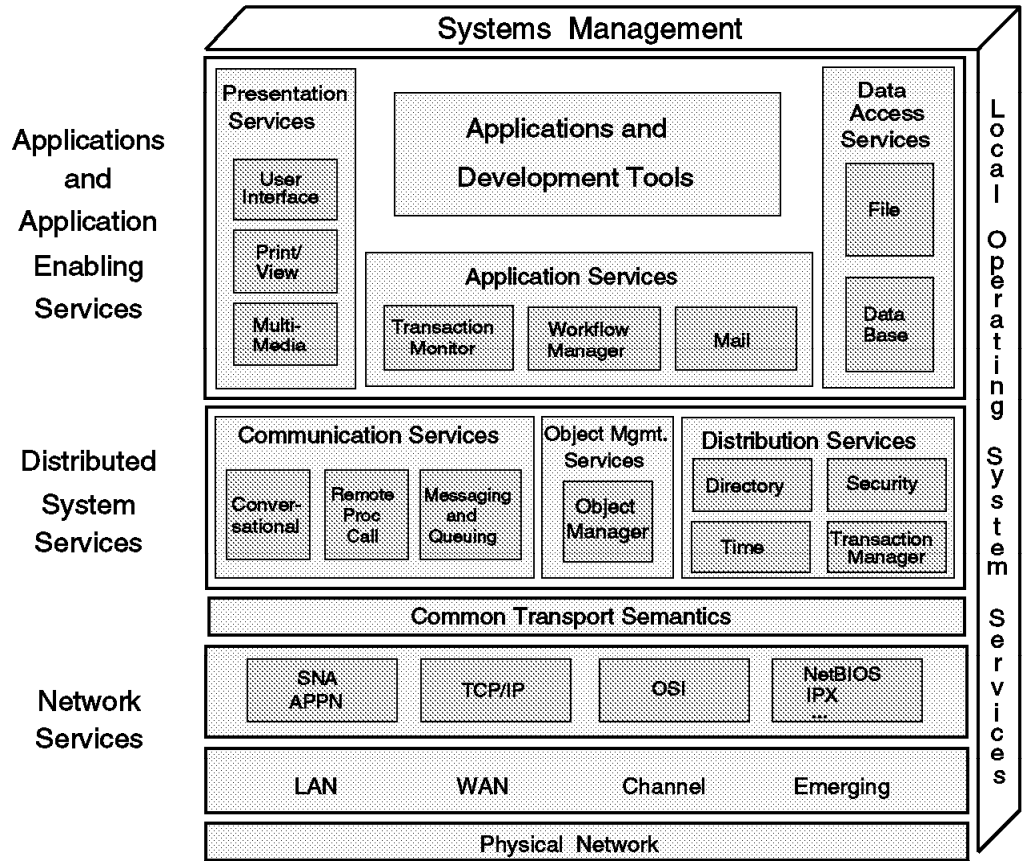


Figure 1. The Open Blueprint

These structures are discussed in considerable detail in *The Library for Systems Solutions, Computing Technology Reference,*. The present document will concentrate on the layer marked *Distributed System Services*, and more specifically, on the area called *Distribution Services*.

1.7 Client/Server Considerations

No book about computing technologies would be complete if it didn't address client/server applications. In their excellent book on client/server computing,[1] Orfali and Harkey speak of the importance of creating a "single system image" of all the services on a client/server network. They refer to this as a *grand illusion* that makes all the servers — no matter how complex the network — appear to the client as one big happy family. The means for creating this illusion are contained in the services and functions that belong to neither the client nor the server, and which are given the name "middleware." To be able to provide a real distributed computing environment, these services must be able to make the network appear transparent to the user. The client server architecture is divided into three layers of services called *Topware*, *Middleware*, and *Bottomware* (see Figure 2 on page 7). If we look at what transparency

[1] Robert Orfali, Dan Harkey, Client/Server Survival Guide with OS/2*, Van Nostrand Reinhold, New York, ISBN 0-442-01798-7.

means, we will see that the directory, naming, and time services fall squarely into the area of middleware, when seen in a client/server environment.

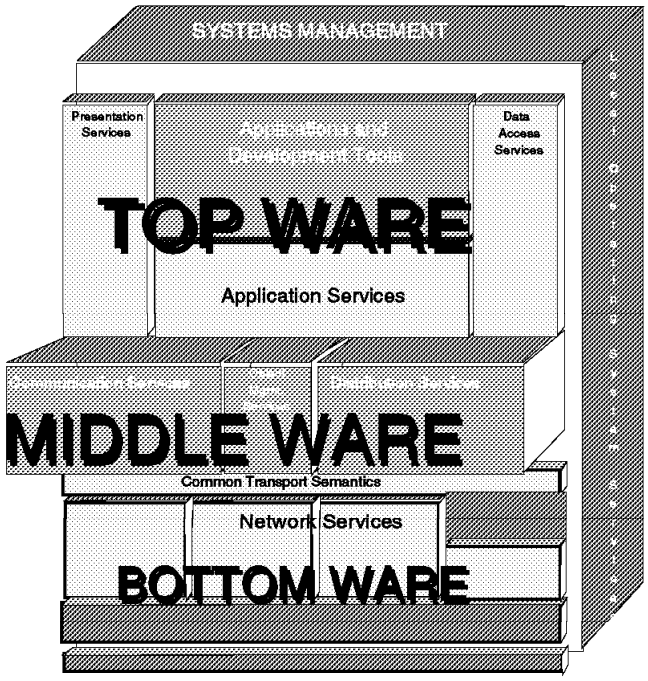


Figure 2. Client/Server Architecture

1.7.1 Network Transparency

"Transparency means fooling everyone into thinking the client/server system is totally seamless. It really means hiding the network and its servers from the users and even the application programmers.

The following are some of the types of transparency this entails:

Location transparency You should not have to be aware of the location of a resource. Users should not have to include the location information in the resource's name. For example, *Machinedirectoryfile* surfaces the name of the server machine. This is a transparency violation.

Namespace transparency You should be able to use the same naming conventions (and namespace) to locate any resource on the network. The whole universe is one big naming tree. This includes every type of resource on any vendor's product

Containment transparency You should be able to move resources on the servers without requiring any changes on the client. For example, you should be able to move a file from one subdirectory to another on the server and make it totally transparent to the client machines.

Logon transparency You should be able to provide a single password (or authentication) that works on all servers and for all services on the network.

Replication transparency You should not be able to tell how many copies of a resource exist. For example, if a naming directory is shadowed on many machines, it is up to the the NOS (network operating system) to synchronize updates and take care of any locking issues.

Distributed access transparency You should be able to work with any resource on the network as if it were on the local machine. The NOS must handle access controls and provide directory services.

Distributed time transparency You should not see any time differences across servers. The NOS must synchronize the clocks on all servers.

Failure transparency You must be shielded from network failures. The NOS must handle retries and session reconnects. It must also provide some levels of service redundancy for fault-tolerance.

Administration transparency You should only have to deal with a single system management interface. The NOS must be integrated with the local management services."[2]

It is middleware that is the prime component in maintaining all these types of transparency. Of the middleware services, directory, naming, and time are actively participating in nearly all of the areas and are the major components for some of them.

[2] ibid. p.218

Chapter 2. Directory Technologies and Standards

In a distributed environment, it is necessary to be able to locate various kinds of entities in a reasonable amount of time. The general term for this kind of locator is directory.

The word “directory” means many things in a computing environment. It can be a list of files limited by some qualifier, as in a DOS directory or subdirectory. That type of directory is a given in the system that includes it and, therefore, not usually the subject of investigation and comparisons. Instead, this document will concentrate on directories proposed and implemented for controlling distributed systems.

There are both proprietary and open directory standards available today. The open standards have been implemented in various forms as part of individual products. Most of these implementations are incomplete ones. This chapter will concentrate on describing the open standards. The proprietary standards and the actual implementation of the open standards will be described in Chapter 5, “Directory, Naming and Time Solutions” on page 117.

This chapter is divided into the following sections:

- Directory requirements.

  This section discusses the requirements that a distributed directory must meet. The evolution of the Internet directory is used to illustrate the challenges that arise when creating a directory.

- DNS naming.

  This section discusses the naming standard developed for TCP/IP which, while not an international *de jure* standard, is very widely used for distributed computing. It is one of the directory standards supported by OSF in its Distributed Computing Environment (DCE).

- X.500 Directory.

  This section discusses the main international standard for directories today. It is the other directory standard supported by OSF in its Distributed Computing Environment (DCE).

- The APPN* Directory.

  APPN (Advanced Peer-to-Peer Networking*) is an enhancement to SNA (Systems Network Architecture) that allows nodes to communicate as peers without the services of an SNA host computer. While it oversimplifies the case, it is useful to think of APPN in the same context as TCP/IP, because they solve many of the same problems.

  APPN uses a sophisticated distributed directory to locate network resources, and it is the directory that will be discussed in this document. It does not contain an in-depth discussion of APPN′s capabilities nor does it provide a comparison with other products, such as TCP/IP. That information should be obtained from the appropriate networking literature.

- The IBM Open Blueprint Directory

  This section discusses briefly the directory standard specified for the Open Blueprint. Since the Open Blueprint is based on the X.500 Directory, detailed information can be obtained by studying that standard.

- OSF DCE

  The Distributed Computing Environment (DCE) that has been developed by the Open Software Foundation, an industry consortium, has attained immense popularity with several vendors — including IBM — producing more or less complete implementations on a wide variety of platforms. The direction taken by OSF in developing DCE is slightly different then that taken by standards bodies in developing their standards. OSF does not create new standards and architectures, but tries to find the “best of breed” products available and adopt them as the DCE standard. Like IBM in its Open Blueprint, OSF has chosen a variation of the X.500 Directory for its directory standard, but it also supports DNS naming.

2.1 Directory Requirements

The more complex and dynamic a distributed environment becomes, the more important it is to have a Directory and Directory Services that are responsive and provide up-to-date information without themselves becoming a performance burden on the network, and the more difficult this task becomes.

A classic example of these considerations is the Internet directory.

2.2 The Internet Directory

The Internet directory is a *de facto* directory standard. It differs from the *de jure* standards in that it was developed to satisfy a concrete requirement for a high-performance distributed directory, capable of handling a fairly high rate of changes. In spite of its “dirty” origins, it has become very pervasive in many different distributed computing environments.

2.2.1 What is the Internet?

The Internet is an outgrowth of a U.S. Department of Defense experiment called the ARPANET (named for the DARPA, the Advanced Research Projects Agency of the Department of Defense), which was to provide a wide-area network to allow government contractors to share expensive and scarce computing resources.

The users of the network quickly found other uses for the network, such as sharing files and software, and – most visibly – exchanging electronic mail.

The TCP/IP group of protocols was developed and became the dominant host networking protocol on the ARPANET. These protocols found wide distribution throughout the academic and research communities by being included in some of the more popular (and low-cost) versions of UNIX. The ease with which these UNIX users could communicate with ARPANET users led to an explosion in the number of host computers connected to the ARPANET.

Deciding the experiment was over, the DARPA began dismantling the ARPANET in 1988. Another network, funded by the U.S. National Science Foundation, called NSFNET, replaced the ARPANET as the Internet backbone.

During the 1970′s, the ARPANET remained relatively small, with only a few hundred hosts connected to it. A single file, HOSTS.TXT, was maintained that

contained all relevant information about those member hosts: it held a name-to-address mapping for every host connected to the ARPANET.

This file was maintained by SRI's *Network Information Center* and distributed from a single host, SRI-NIC . ARPANET administrators usually e-mailed their changes to the NIC, and would periodically use FTP to retrieve the most current HOSTS.TXT. HOSTS.TXT was recompiled once or twice a week to reflect the new changes. As the ARPANET grew, this scheme became unworkable. HOSTS.TXT grew proportionally with the number of ARPANET hosts. Moreover, the traffic generated by the update process grew even faster: every additional host meant not only another entry in HOSTS.TXT, but potentially another host retrieving the file from SRI-NIC.

When the ARPANET moved to using TCP/IP, the network population exploded. This resulted in three major problems with HOSTS.TXT:

- Traffic and load.

  The network traffic and corresponding processor load on SRI-NIC became unsustainable.

- Name collisions.

  No two hosts in HOSTS.TXT could have the same name. However, while the NIC could assign addresses in a way that guaranteed uniqueness, it had no authority over host name. There was nothing to prevent someone from adding a host with a conflicting name and breaking the whole scheme. Someone adding a host with the same name as a major mail hub, for example, could disrupt mail service to much of the network.

- Consistency.

  Maintaining consistency of the file across an expanding network became harder and harder. By the time a new HOSTS.TXT could reach the farthest limits of the ARPANET, a host across the network had changed addresses, or a new host, that you wanted to reach, had sprung up.

The problem was that HOSTS.TXT did not scale well.

The ARPANET governing bodies chartered an investigation into a successor for HOSTS.TXT. Their goal was to create a system that solved the problems inherent in a unified host table system. The new system should allow local administration of data yet make that data globally available. The decentralization of administration would eliminate the single-host bottleneck and relieve the traffic problem. And local management would make the task of keeping data up-to-date much easier. It should use a hierarchical namespace to name hosts, thus assuring uniqueness of names.

The result of the investigation was the Domain Name System of the Internet (as set down in RFCs 882 and 883, later superseded by 1034 and 1035), but the problem definition included the major problems that would have to be addressed by any directory to be implemented in a large, dynamically changing, distributed environment.

2.2.2 Importance of Internet

The Internet is a collection of computer networks that cooperate to provide communications services such as electronic mail, file transfer, and bulletin boards. The Internet is significant in a discussion of global naming for several reasons: it has a wide constituency, including U.S. Government agencies, educational institutions, and commercial enterprises; it is international in scope, with links to many European countries as well as other countries; and it is growing rapidly— from approximately 200 connected host computers in 1980 to over a million today.

2.2.3 Internet Naming

The Internet naming system is called the *Domain Name System* (DNS). Before the DNS was introduced, the name and address of every host computer in the Internet were maintained in a centralized host table. With rapid expansion that crossed international boundaries came the requirement for an adequate global naming system. The DNS currently satisfies this requirement for Internet users.

2.2.3.1 IP Addresses

DNS naming is closely related to addressing of networks within the Internet. Local networks connected to the Internet are distinguished by unique Internet Protocol (IP) network numbers. The Transmission Control Protocol (TCP) and the Internet Protocol (IP) are foundational standards supporting open transfer of information among networks connected to the Internet. IP is a lower-level protocol that defines the packets or datagrams that traverse the Internet, and IP provides some functionality for routing, storing, and forwarding packets. TCP is a higher-level protocol that establishes virtual circuits to support reliable transmission of messages. For example, TCP includes functionality to acknowledge receipt of data and retransmission of lost or damaged data.

IP numbers or addresses are 32-bit numbers assigned to host computers connected to the Internet. This 32-bit number is divided into four bytes separated by periods, and is typically represented in decimal— for example, 128.5.0.0. Currently IP numbers have the following structure:

- In Class A addresses, the first byte is the network portion and the remaining bytes are the local portion.
- In Class B addresses, the first two bytes are the network portion and the remaining bytes are the local portion
- In Class C addresses, the first three bytes are the network portion and the remaining byte is the local portion.

Class is indicated by the value in the first byte. For example, in class C addresses the first byte is in the range 192 - 223. Class C IP addresses are typically assigned to local area networks.

Central registration of IP addresses is provided by the Defense Data Network Network Information Center (DDN NIC). The DDN is a military network within the U.S. Department of Defense that historically is related to the now-obsolete ARPANET, which was the original basis of the Internet. DDN NIC can delegate regional registration responsibility to other organizations as necessary—for example, the RIPE (Reseaux IP Europeens (InterEUNet database)) Network Coordination Center in Amsterdam coordinates IP network number registration in Europe. DDN NIC only assigns the network portion of the IP address; the local portion is assigned by the local network administrator.

2.2.3.2 Name Mapping

The primary purpose of the DNS is to provide mapping between user-oriented names and IP addresses within the Internet namespace. Conceptually the DNS is a hierarchy of names which replaces the original centralized table that once identified all entities connected to the Internet. In a DNS name segments are separated by periods, with the highest-level (that is, the most general) segment occurring in the rightmost position. The leftmost segment is the most specific portion of a DNS name. DNS domains are distinct from networks. Domains are hierarchically organized administrative entities that maintain sub-namespaces, and may contain one or more networks. A domain may correspond to a single network, in which case the domain name and the network IP address are analogous. But this case is not normative.

The highest-level domains in the DNS can be divided into two groups: (1) domains based on country, and (2) domains based on function.

Highest-level domains based on country are identified by two-letter country codes assigned in ISO 3166. The U.S. domain is represented by “us.” In the U.S., Internet names within the U.S. country domain tend to be organized geographically. A typical usage can have the following structure:

“local node identifier.city name or abbreviation.state name or abbreviation.us”

The other highest-level domain names are not assigned geographically. They are assigned based on the function or purpose of the entities participating in the Internet. The highest-level functional domains are shown in the following list:

COM Commercial enterprises

EDU Educational institutions

MIL U.S. military organizations

GOV Non-military U.S. Government entities

NET Network information centers and other Internet functional components

ORG Non-profit and miscellaneous organizations

INT International organizations (for example, NATO).

The DNS is implemented as a system of cooperating databases, with each individual database storing and maintaining a small portion of the overall DNS set of names. DNS database functionality is based on a client-server model in which *domain name resolvers* or clients issue queries to domain name servers. Locating a DNS node requires mapping the name to an IP address. Depending on the information stored in each name server, resolving a DNS name may require a single query or a series of queries to find a server capable of mapping the name to an IP address.

Information returned from a DNS query consists of *Resource Records*. The following list shows illustrative examples of Resource Records:

Resource Record Type Information Provided

NS (Name Server) Name of host providing name service

A (Internet Address) IP address

MX (Mail Exchanger) Site receiving domain mail.

For e-mail applications of the DNS, a user name is appended to the DNS name of the host that provides mail service to the user. The @ symbol separates the user name from the host name, for example:

wilcox@stutvm2.vnet.ibm.com

2.2.3.3 Directory Services

Work is in progress toward implementing directory services in the Internet based on the OSI X.500 model (see 2.3, “The X.500/ISO 9594 International Standard” on page 15). Working groups of the Internet Engineering Task Force (IETF) are studying ways to add this functionality to the Internet. Currently the Internet maintains a directory service referred to as “WHOIS.” This is a centralized, non-distributed directory that offers information about entities connected to the Internet. It should be noted that the WHOIS directory is oriented toward support of electronic mail; for example, it provides information about users, hosts, and domains. In contrast, the X.500 series of recommendations in intended to provide a general directory service for “objects” of any kind.

The importance of directory services for large heterogeneous networks, such as the Internet, is illustrated by current efforts to help Internet users obtain information about entities connected to the Internet. At least five major development projects have taken place to help Internet users discover what information is available. An example is the “Internet Gopher”— a system of servers developed by a U.S. university that offers access to Internet data on news, weather, books, and other topics.

2.2.4 Relationship of the Internet to OSI

In certain respects, the Internet has common objectives with the OSI model— for example, to provide interoperability between distributed heterogeneous systems. The OSI reference model describes a seven-layer stack of protocols enabling communications between open systems. The Internet protocol suite contains several communications protocols, including the following:

- TCP (Transmission Control Protocol)
- IP (Internet Protocol)
- ICMP (Internet Control Message Protocol)
- FTP (File Transfer Protocol)
- SMTP (Simple Mail Transfer Protocol)
- SNMP (Simple Network Management Protocol)
- UDP (User Datagram Protocol)
- Telnet Protocol.

When compared to the seven-layer stack of the OSI model, the TCP/IP family of protocols can be considered to constitute a four-layer stack:

- Application layer
- Transport layer
- Internet layer
- Network interface layer.

In terms of the OSI model, IP is considered to operate at the OSI network layer, and TCP is considered to operate at the OSI transport layer. TCP most resembles class 4 OSI transport service.

2.2.5 Internet Trends and Directions

2.2.5.1 Future of DNS Naming

DNS naming is dependent on the future of IP addressing, on which it is based. The current design of IP addressing is reaching its upper limits, especially with respect to Class B addresses, because of the rapid growth of the Internet. The IETF is at work on schemes to enlarge the DNS namespace in a way that is upwardly compatible with present Internet usage.

2.2.5.2 Integration with OSI

OSI and TCP/IP represent different approaches to a common problem: universal interoperability in a heterogeneous world. Historical and philosophical differences distinguish these two approaches, but the increasingly urgent requirement for consistent interoperability is generating more attention to the feasibility of integrating the two. For example, the IETF maintains working groups that study the possibility of integration with OSI with regard to directory services. In the OSI world a recent proposal was made in Europe to broaden the scope of OSI conformance testing, as defined in ISO 9646, to include testing of mixed protocol stacks containing elements from the set of OSI standards and from the TCP/IP suite of protocols.

2.2.5.3 Naming of Objects

For the present time, the Internet naming system appears to be adequately supporting the requirement to identify Internet host computers, users, and other network entities. Historically, the Internet naming system grew rapidly to provide unique identifiers for network participants. There does not appear to be a design requirement for the DNS to provide naming of arbitrary information objects. In this respect, the Internet naming system has a different purpose and scope from more general approaches to naming, such as ISO ASN.1 object identifiers, ISO public text, and ISO structured names.

2.3 The X.500/ISO 9594 International Standard

The CCITT X.500 series of recommendations form the directory standard that seems to be attracting the most attention by implementers at this time. See Chapter 5, “Directory, Naming and Time Solutions” on page 117 for information on directory implementations. It is identical to the standard documented by ISO/IEC 9594. Common, everyday usage seems to favor the CCITT name, X.500.

While there doesn′t seem to be a complete implementation on the market at the time of this writing (mid 1994), a number of vendors are implementing parts of the standard, and others are looking to do the same.

X.500 is an OSI Management standard (see Appendix A, “OSI Reference Model” on page 123) at the application layer specifying principles, protocols, and procedures for storing and retrieval of information about “objects.” It provides an information structure model, specifies protocols for communication about directory information, and specifies procedures that allow directory information to be distributed among several systems, including procedures for navigation to the system containing the information to be accessed.

The directory service is intended to provide functions, such as user-friendly naming, name-to-address mapping, and dynamic assignment of names. Naming of objects in X.500 is based on the idea of collections of attributes which locate objects within a directory information tree (DIT) contained in a directory information base (DIB). Both of these concepts will be explained in greater detail later in the chapter.

Objects can be anything that can be given one or more unambiguous names and for which it is of interest to store information for later retrieval.

The directory is not intended to be a general purpose database, but has mainly been developed for storing information about objects relevant in the telecommunications arena, such as application entities, persons, files, distribution lists, and so on. The information stored about objects is typically information relevant for communication with or about those objects. The information is expected to be relatively static, and not subject to rapid change. This information should be location independent. It is suitable for data such as names, addresses, and telephone numbers, but unsuitable for data, such as the number of users signed on to an application, or the name of the next link in a communication path.

Even with its current deficiencies, the X.500 standard is functionally rich and flexible. But this flexibility, particularly in the data structure means that just pointing two directory systems at each other does not guarantee that they will interoperate. Systems that wish to interoperate must have a common understanding of the data elements (which are not mandated by the standard) as well as the dialogue protocols (which are). Interest groups are beginning to define areas of common understanding, but much work still needs to be done before X.500 can be fully exploited. Just remember that these understandings for directory interoperability are required whatever protocols are used; X.500 is no different in this respect.

The X.500 standard includes the following characteristics:

- Decentralized management
- Searching capabilities
- Single global namespace
- A structured framework for the storage of information.

The 1988 version of X.500 specifies four models to define the Directory Service:

- The Information Model
- The Functional Model
- The Organizational Model
- The Security Model.

2.3.1 Directory Overview Model

The function that maintains and communicates directory information is called the Directory System Agent (DSA). In OSI terms, a DSA is an application process. An open system may contain any number of DSAs. DSAs communicate with each other using an OSI application layer protocol called the Directory Service Protocol (DSP).

Systems without directory information can access the directory information by establishing an association with a DSA. The function performing that access is called a Directory User Agent (DUA). An open system may have any number of DUAs. DUAs communicate with DSAs using an Application Layer protocol called the Directory Access Protocol (DAP).

The communication between a DUA and a DSA is always initiated by the DUA. The DSA will never send anything unsolicited to a DUA. Therefore, an inoperative DUA, in contrast to an inoperative DSA, only affects the local user. DUAs do not communicate with each other.

The collection of DSAs is the Directory, which provides DUA access to directory information independent of the location of that information. The information is kept in the Directory Information Base (DIB), using any suitable database technique. As a special case, the Directory may consist of just a single DSA; such a non-distributed Directory is called a centralized Directory.

When a DUA sends a request to a DSA to access directory information, and that DSA contains none or only part of the information in question, it can basically operate in several modes:

- It can forward the request to one or more other DSAs supposedly more capable of handling the request, and relay the received reply or replies back to the DUA. This mode of operation is called chaining mode, or where chaining is performed to more than one DSA simultaneously, multicasting.
- The DSA can return to the DUA the name and the presentation address of the DSA or DSAs recommended to access. This mode is called referral mode. Referral mode also works between DSA and DSA.

2.3.2 Directory Information model

An object is represented by a Directory entry, which contains information about the object.

An entry has to be completely maintained by a single DSA (by standards as of 1988). Copies can exist in other DSAs, but the method by which these copy entries are initiated in other DSAs and how they are maintained is outside the 1988 standard, (see 2.3.10, “Areas Excluded from the 1988 Directory Standard” on page 38). These copies can be used to improve performance, but the user can insist on using the master entry to ensure the currency of the information. Updates can be made only to the master entry.

The complete information maintained by the Directory is called the Directory Information Base (DIB).

When communicating about objects, it is generally important to know what kind of object it is. Object classes are defined, each representing objects of a specific type (for example all application entities belong to the Application Entity object class). Figure 3 on page 18 gives a list of some typical object classes.

- An object is represented by a directory entry
- This entry holds information (attributes) about the object
- Directory Information Base (DIB) = Collection of all directory information
- Examples of objects are:
  - persons
  - application entities
  - files
  - distribution lists
  - groups
  - organizations
  - organizational units

Figure 3. Directory Entry and Objects

2.3.2.1 Directory Information Tree (DIT)

The Directory Information Model provides a hierarchical structure, in that entries "belong to" or are "subordinate" to other entries. This model works well for many objects; for example, cities are subordinate to states, which are in turn subordinate to a country. However, it can result in multiple entries for the same real life object (location person and organizational person), and result in difficulties in storing multiple relationships.

To store information about objects in the Directory, objects have to be named. A name has to be unique in the sense that it denotes exactly one object entry.

A directory entry name consists of a sequence of distinct name components, representing each level of the naming hierarchy. The last component of an entry is called the Relative Distinguished Name (RDN) for that entry. The full Distinguished Name (DN) is the concatenation of all RDNs from the root down to the entry in question.

A directory entry name has to be unique across all interworking directory systems. Since these systems may span many different countries and organizations, agreements have to be reached on common naming authorities for the names. Entry names can, in principle, be allocated by a hierarchy of naming authorities, each level of authority being responsible for a single name component, and the resulting hierarchical naming structure can be represented by a naming tree, like the one shown in Figure 4 on page 19. A name component allocated by a naming authority need only be unique with respect to that authority or hierarchy level.

The choice of naming authorities is not necessarily a simple matter. There may be competing authorities at the same level, with different characteristics, such as administrative simplicity and price. Much thought is being given by implementation groups, such as NADF, to the selection of the most appropriate naming authorities. Different communities may choose different authorities which may cause problems if those communities eventually have to interwork. For example, one authority may choose to define location entries according to census names, another according to postal service names. High-level naming authorities are typically national authorities, such as PTTs. Organizations act as their own naming authority for lower level entries, such as departments.

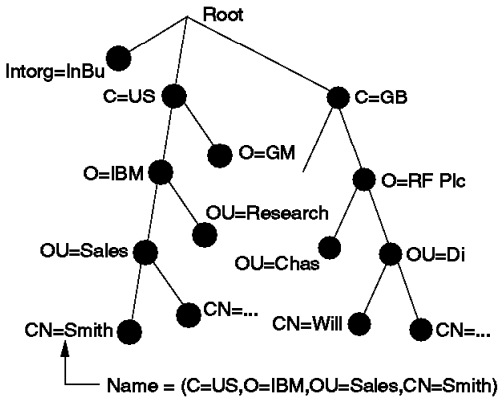


Figure 4. Directory Information Tree (DIT)

X.500 defines (in X.521) a DIT structure which will have widespread applicability. This structure is used below to demonstrate X.500 concepts. Other structures are permitted by the standard. The first level of entries below the root are countries and international organizations. (Country names are taken from ISO 3166 — "The international codes for country names.")

In the example shown, the next level is the organization, such as company or government agency. The full name of an organization is the country name concatenated with the organization name component (for example, C=US, O=IBM).

The next two levels in the example are the organizational unit and person. The latter is identified by its Common Name. The corresponding names could be (C=US, O=IBM, OU=Sales) and (C=US, O=IBM, OU=Sales, CN=Wilcox), respectively. Note that X.500 names are typed, that is, they are specified as a series of "type=value" pairs.

Each vertex of the naming tree represents an allocated name. Each entry of the DIT has one unique and unambiguous name; that is, an entry has only one name, and that name is different from any other entry name.

The DIT can be distributed among DSAs in any way. The entries held by a DSA can, in principle, be an arbitrary collection of entries and do not have to be closely related to each other. In real life, one would expect that a DSA contain groups of closely related entries, but the intention of the Directory standard is to cope with any distribution of entries.

Several independent directories, that is, several independent DITs, may be created. However, if the names of the objects represented by these directories conform to the same naming conventions and authorities, they could be merged into one directory, if they have otherwise compatible implementations.

2.3.2.2 Object and Alias Entries

Each object is represented in the directory by one principal entry, called the object entry. This entry holds the information related to that object. It is often convenient to provide an object with more than one name. This is achieved through the use of special entries, called aliases. These entries are uniquely named in the usual way but point to the object entry. The alias entries are

illustrated on Figure 5 on page 20. Each alias entry thus provides an alternative name for an object. An object can have more than one alias entry. The resolution of alias names to “real” names is done by the DSA.

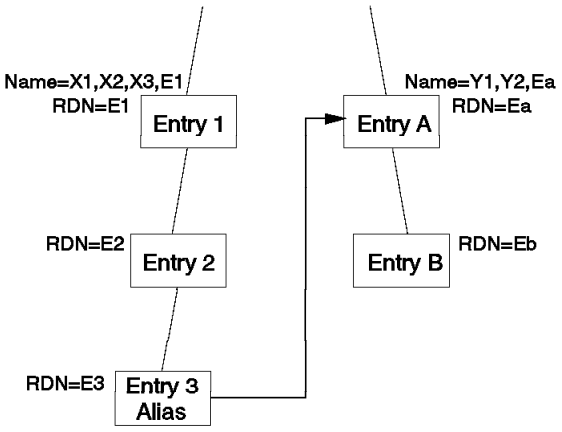


Figure 5. Alias Entries and Alias Names

2.3.2.3 Entries and Entry Attributes

The Directory holds information about an object as a number of entry attributes. Figure 6 illustrates how attributes, like country telephone code, address, and so on are associated with entries.

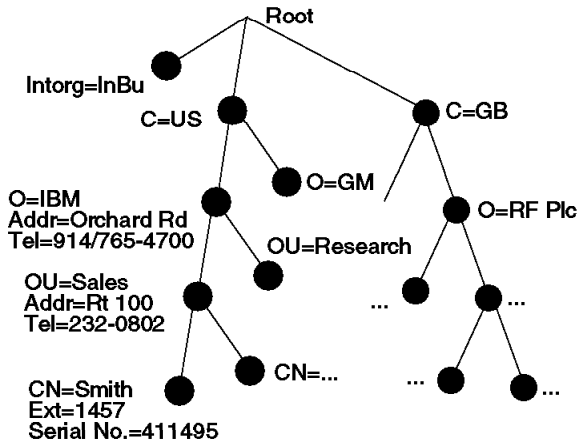


Figure 6. Directory Entries and Attributes

Note that the same attribute type, possibly with different attribute values, can be held by different object entries. This is illustrated in the example by the “Addr” attribute having one value at the organization level and another value at the department or organizational unit level.

2.3.2.4 Entry Structure Model

The main function of the Directory is to communicate the attributes of directory entries. Attributes are carried as part of the protocol exchanges. It is therefore necessary for directory systems to agree on the syntax of the attributes. The Directory is not normally concerned with the semantics of an attribute; that is the responsibility of the application or user. There are some specific attributes that affect the operation of the Directory, for which the semantics are defined.

Figure 7 shows the model for an entry and its attributes, where an attribute consists of an attribute type and one or more attribute values.

The presentation address attribute of an application entity entry is an example of an attribute having only one value, while a telephone number attribute of a person may have several values. No semantics can be associated with the order of the values, so the first telephone number in a list cannot be assumed to be the preferred one. If the concept of preferred telephone number were important in an implementation, a special attribute of “Preferred Telephone Number” would be required.

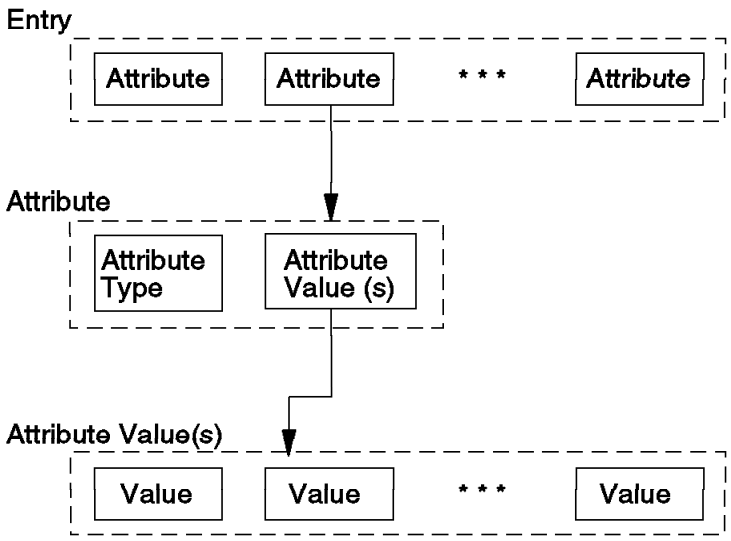


Figure 7. Directory Entry Structure

2.3.2.5 Relative Distinguished Names

The Relative Distinguished Name (RDN) identifies an entry relative to its immediate superior entry in the tree, and consists of one or more type/value pairs. While there may be special cases where the use of multi-attribute RDNs is justified, it is normally possible and desirable to choose the RDN structure based on a single attribute. There are some advantages if RDNs (and hence the full Distinguished Name) can be accurately constructed from readily available information. This allows direct (and efficient) access to specific entries. If the

RDN cannot be readily constructed, then the available information must be used to filter a wider search operation based on non-naming attributes, which is generally less efficient.

The choice of naming attribute may not be a trivial or simple task, however. As an example, if the attribute “Person_Name” has been chosen to identify people within departments, this would normally be sufficient to provide uniqueness. However, occasionally more than one person in a department will have the same Person_Name (for example “John Smith”), additional information has to be added to ensure uniqueness. There are several different ways to solve this problem (see also Figure 8):

- Adding another attribute (such as “title”) would provide uniqueness, but would then be required for every person entry. Interoperating directories would have to know of and work with this RDN structure.
- Adding a modifier to the value of one or more entries, for example, “John A. Smith.” This might result in directory operations missing one or both entries.
- Adding a special attribute, “See_also,” to both entries may help applications and users to recognize that the problem exists, but only if they look for it.
- A neutral entry for “John Smith” that points to both real entries could be used, but the applications and users would have to be aware of this possibility.

Relative Distinguished Name (RDN) for Entry:

- Set of (attribute type, attribute value) pairs
- Unique within scope of immediately superior entry.

Example of single attribute:

RDN = (Person_Name, John Smith)

Example of multi-attribute:

RDN = ((Person_Name, John Smith),(Title, Clerk))

or

RDN = ((Title, Clerk),(Person_Name, John Smith))

Figure 8. Relative Distinguished Names

2.3.3 Schema, Objects, Rules

The way in which the directory is put together is called the schema. It is defined by rules and definitions for various components.

2.3.3.1 Selected Attribute Types

A number of generally usable attribute types have been defined by the Directory standard and are listed in Figure 9 on page 23 and Figure 10 on page 24.

More specialized attributes can be defined. Organizations, vendors, and so forth, can define their own attribute types and use them to create attributes for storing and communicating specific information. Likewise, attribute types can be defined by other standards. X.400 has defined a number of attribute types for storing X.400 specific information in the Directory.

2.3.3.2 Attribute Syntax

As attributes are carried by the protocol exchanged between DSAs and DUAs, the syntax of attributes has to be fully specified and understood. A number of generally usable attribute syntax types are defined by the Directory standard. The defined syntaxes cover widely used syntaxes, such as “printable string” and “integer,” to more specific syntaxes, such as “distinguished name” and “telex number.” Directory administrators can also define private syntaxes to accommodate private attributes, but these would not be generally accessible by other DSAs and DUAs.

- System Attribute Types
  - Object Class
  - Aliased Object Name
  - Knowledge Information
- Labelling Attribute Types
  - Common Name
  - Surname
  - Serial Number
- Geographical Attribute Types
  - Country Name
  - Locality Name
  - State or Province Name
  - Street Address
- Organizational Attribute Types
  - Organization Name
  - Organizational Unit Name
  - Title
- Explanatory Attribute Types
  - Description
  - Search Guide
  - Business Category
- Postal Addressing Attribute Types
  - Postal Address
  - Postal Code
  - Post Office Box
  - Physical Delivery Office Name
- Telecommunications Addressing Attribute Types
  - Telephone Number
  - Telex Number
  - Facsimile Telephone Number
  - X.121 Address
  - ISDN Address
  - Registered Address
  - Destination Indicator

Figure 9. Selected Attribute Types - 1

- Preference Attribute Types
  - Preferred Delivery Method
- OSI Application Attribute Types
  - Presentation Address
  - Supported Application Context
- Relational Attribute Types
  - Member
  - Owner
  - Role Occupant
  - See Also
- Security Attribute Type
  - User Password
  - User Certificate
  - CA Certificate
  - Authority Revocation List
  - Certificate Revocation List

Figure 10. Selected Attribute Types - 2

2.3.3.3 Object Classes

The Directory can be used for many different purposes. A message handling system (MHS) user may wish to map the directory name of an originator/recipient to its O/R name, requiring MHS-specific object class entries to be present in the Directory; an FTAM user may want to map file names onto locality information, requiring FTAM-specific object class entries to be present, and so on.

The naming structure allows all such object class entries to be part of the same DIT, but some DSAs may need to support only a limited number of object classes. “Specialized” DSAs are possible, like a DSA having entries only for objects related to a manufacturing floor, for example.

For two or more open systems to communicate meaningfully about objects, it is necessary for those open systems to have a common understanding of:

- What object classes are subjects for communication
- The name structure of the objects
- The attributes associated with the objects.

A basic set of object classes with general applicability independent of the specific use of the Directory has been standardized by the Directory standard (ISO/IEC 9594-7 and X.521 — “Selected Object Classes.” See Figure 11 on page 25).

- Top
- Alias
- Country
- Locality
- Organization
- Organizational Unit
- Person
- Organizational Person
- Organizational Role
- Group of Names
- Residential Person
- Application Process
- Application Entity
- DSA
- Device

Figure 11. Selected Object Classes

Other standards may define object classes for specific use. As an example, MHS-specific object classes are defined by ISO/IEC 10021-2 (X.402-1988) — "MOTIS (Message Handling Systems) — Overall Architecture." Object classes can also be defined for specific purposes by organizations outside the standards community. For example, a private organization can define its own specific object classes.

The definition of an object class includes:

- Assignment of object-identifier
- Indication of the class of which it is a subclass
- Indication of mandatory attributes
- Indication of optional attributes.

2.3.3.4 Subclasses

Object classes may be defined as a subclass of a previously defined and more general object class as depicted in Figure 12. This simplifies administration and definition of object classes, as a subclass inherits the attribute definitions of its "parent" object class.

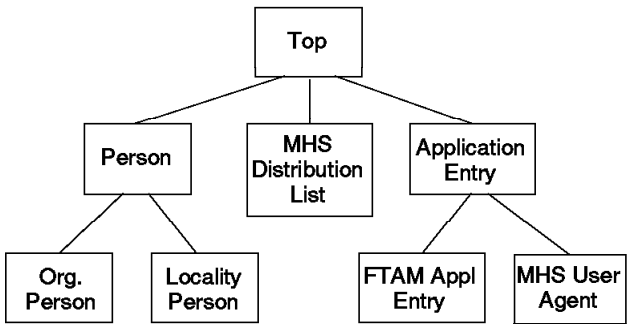


Figure 12. Subclasses of Object Classes

2.3.3.5 DIT Structure Rules

DIT structure rules define for each object class:

- Object class for immediate superior entry
- Object class for immediate subordinate entry
- Naming attributes.

An object class may be allowed at several points in the hierarchy; for example, organization class could be allowed directly under the root (for international organizations), and under country (for national organizations). Figure 13 illustrates these structure rules.

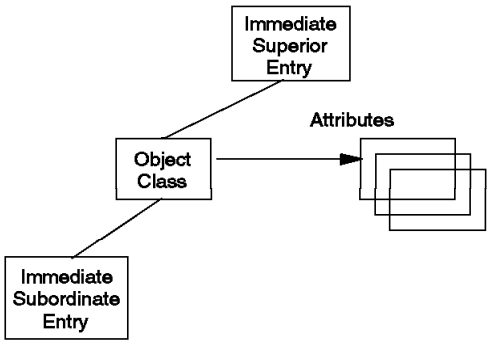


Figure 13. DIT Structure Rules

2.3.4 Directory Schema

Putting the definitions of Attribute Syntax, Attribute Types, Object Classes, and Structure Rules together produces what is known as the Schema. Figure 14 illustrates the structure of the schema.

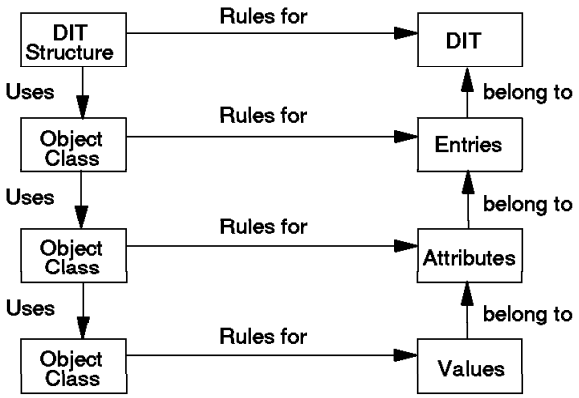


Figure 14. Directory Schema

2.3.5 Directory Service

This section describes the operations that can be requested by a DUA to be performed on the directory data, and this is summarized in Figure 15.

Read Operation Types:

- Read (Entry)
- Compare (Attributes)
- Abandon

Search Operation Types:

- Search
- List (subordinates)

Modify Operation Types:

- Add Entry
- Remove Entry
- Modify Entry
- Modify RDN

Figure 15. Directory Service Operations Overview

2.3.5.1 Abandon

Abandon is used to remove outstanding directory operations, where the application is no longer interested in the results. It may be used only on interrogative operations such as Read, List, Search, and Compare. Figure 16 illustrates how to it is done.

Purpose: To abandon previous directory operation

Input arguments:

- Invoke-Id: the unique identifier for the previous operation

Results:

- Abandoned error on the previous operation

Figure 16. Abandon

2.3.5.2 Read

To perform a Read operation, the full distinguished name, or an equivalent alias must be provided. Figure 17 on page 28 illustrates how the Read operation.

Purpose: To extract information from a specific object entry

Input Arguments:

- Distinguished name (or alias) of object entry
- Granularity of information to be returned:
  - All attributes – type and value(s)
  - All attributes – type only
  - Selected attributes – type and value(s)
  - Selected attributes – type only

Results:

- distinguished name of object entry (alias resolved)
- entry information (attribute types and values)

Figure 17. Read

2.3.5.3 Compare

Compare can be used to validate the purported value of an attribute without revealing the actual value of the attribute, for example, to compare a password. Only equality match or no-match is provided; otherwise, the actual value could be inferred from a number of attempts. See about the Compare operation in Figure 18.

Purpose: To compare an attribute value of a specific object entry

Input Arguments:

- Distinguished name (or alias) of object entry
- Type and value of attribute

Results:

- Distinguished name of object entry (alias resolved)
- Match or no-match

Figure 18. Compare

2.3.5.4 List

The List operation returns the relative distinguished name information for directly subordinate objects. The application, with or without the help of a user, can then explore these entries with further directory commands, if necessary. For example, the List operation could list all departments within an organizational unit, as shown on Figure 19 on page 29. If the number of departments or the naming attribute was all that was required, then no further operations are needed. If other attributes, such as the mailing address, or lower level information such as the list of people working in a department were required, then further directory operations would be necessary.

Purpose: To list all immediate subordinates of a specific object entry.

Input arguments:

- Distinguished name (or alias) of object entry

Results for each Subordinate:

- RDN of subordinate entries
- Whether object or alias entry

Figure 19. List

2.3.5.5 Search

Search is used where the distinguished name is not known, but some other attributes are. In general, the more information provided, the more efficient the operation. The efficiency will also depend on the particular database and indexing technology used by the DSA implementation. Complex filters can be constructed, if required. Various arguments to a search operation can be found on Figure 20.

Purpose: To find and retrieve information from entries fulfilling certain requirements.

Input Arguments:

- Distinguished name (or alias) of base object entry
- Depth of search:
  - Base object entry only
  - Immediate subordinate object entries only
  - Whole subtree starting at base object entry
- Filter item (attribute condition):
  - Equality
  - Substring
  - Greater or equal
  - Present
  - Approximate match

  or multiple filter items combined with logical operators (and, or, not)
- Granularity of information to be returned:
  - All attributes – type and value(s)
  - All attributes – type only
  - Selected attributes – type and value(s)
  - Selected attributes – type only

Figure 20. Search

On Figure 21 is shown the results from a successful search.

Results:

- Distinguished name of base object entry
- Information for each entry matching filter conditions:
  - Distinguished name of entry
  - Entry information

Figure 21. Result from Search

2.3.5.6 Modification Operations

The following operations which modify entries have some additional conditions attached:

- The operation is performed on the master entry. The 1988 standard assumes that some proprietary method will be used to propagate such master changes to copy entries (if copies exist).
- The distinguished name must be the “real” name and not an alias.

Add Entry Figure 22 shows the Add Entry operation.

Purpose: To add entry in same DSA as superior entry.

Input Arguments:

- Distinguished name of entry (alias not allowed)
- Attributes (types and values)

Figure 22. Add Entry

The directory will reject any entry that does not conform to the schema. For example, an attempt to add an object entry at an inappropriate level in the hierarchy or to add one that contains undefined attributes will be rejected.

Alias entries are created using this operation, one of the attributes being the distinguished name of the object entry being aliased. Note that the directory does not check that the referenced object entry actually exists.

Remove Entry Figure 23 shows the Remove Entry operation.

Purpose: To remove an existing entry

Input Arguments:

- Distinguished name of entry (alias not allowed)

Figure 23. Remove Entry

The entry to be removed must be a leaf entry, that is, it must have no subordinate entries.

Modify Entry Figure 24 on page 31 shows the Modify Entry operation.

Purpose: To add or delete attributes or attribute values not affecting the RDN.

Input Arguments:

- Distinguished name of entry (alias not allowed)
- List of modification requests:
  - Add attribute
  - Remove attribute
  - Add value
  - Remove value

Figure 24. Modify Entry

Modify Entry may only be performed on the master entry, and the DN must be explicitly defined. It may not modify an attribute value that is part of the RDN. It may modify non-leaf entries.

Modify RDN Figure 25 shows the Modify RDN operation.

Purpose: To change relative distinguished name of leaf entry.

Input Arguments:

- Distinguished name of entry
- New RDN
- Whether to add or replace RDN components

Figure 25. Modify RDN

The 1988 standard only allows the RDN of leaf entries to be changed; changing the RDN of a non-leaf entry (and therefore the DN of all its subordinates) must be built up by creating the new entry, copying the individual entries of the subtree across, one by one, and deleting the old tree (from the bottom up).

2.3.6 Distributed Operations

When a DSA receives a request for accessing some entry information and the DSA holds none or only part of the information in question, it can continue to operate in referral or chaining mode.

A DSA may hold only part of the information for a search that spans entries across several levels or across a wide geographic area. The DSA knows from its "knowledge information," (see 2.3.6.3, "Knowledge Information" on page 34), which other DSAs hold more of the tree where information may be found. For example, a search for the SMITHs in UK may involve a local search and referral to other DSAs.

When chaining, the DSA passes the operation to one or more DSAs on behalf of the DUA. When referring, the DSA passes the names of other DSAs back to the enquirer. If the enquirer was another DSA (acting in chaining mode), that DSA can either pass the referral back up the chain or peruse the referral itself.

A special way of chaining is called multicasting or parallel chaining. A DSA decides in this case to forward the request simultaneously to two or more DSAs. There is also the possibility for DSAs to implement referrals between one another.

2.3.6.1 Chaining

In Figure 26, DSA A may have some data or none, but knows that DSA B has more or knows where to find it. The request is passed, in turn, by each DSA to other DSAs that may have more information. The results are merged by each DSA and passed to the previous DSA in the chain. The shading gradations indicate the accumulation of directory information as it is merged from one DSA to the next, starting at DSA D. The merged information is then passed from DSA A to the requesting DUA.

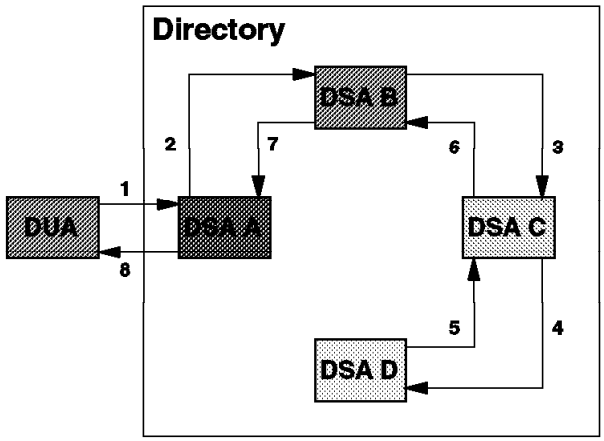


Figure 26. Serial Chaining

In the example in Figure 27, DSA A knows that both DSAs B and D may have information to satisfy the request, so it passes it on to both. DSA A merges the results from DSAs B, D, and itself before returning them to the requesting DUA.

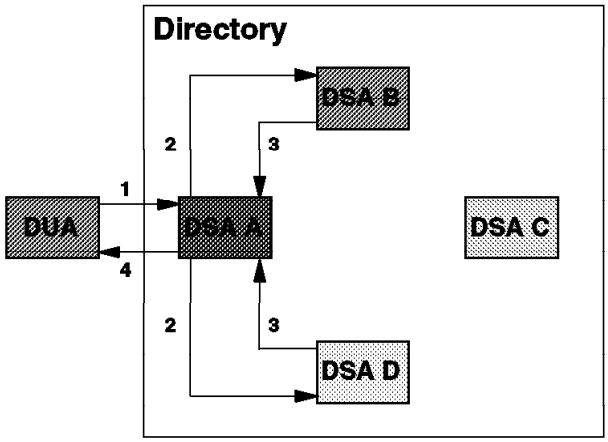


Figure 27. Parallel Chaining or Multicasting

2.3.6.2 Referrals

In the example in Figure 28, DSA A believes DSA B has better information, but instead of passing the request itself, it “refers” the DUA to DSA B. In the example, DSA B does not have the information (or has only part of it) and likewise refers the DUA to DSA D, which is able to satisfy the request. The shading of the rectangles representing DSA B and DSA D indicates that each of them contains part of the sought-for information, but that the total information is finally only present in the DUA.

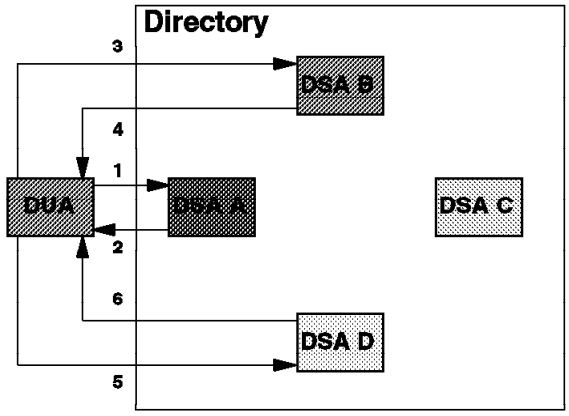


Figure 28. Referrals Pursued by DUA

In the example in Figure 29, DSA A has chained the request to DSA B, which has referred DSA A to DSA D. Instead of passing the referral back to the DUA, DSA A has resolved the referral itself by chaining the command to DSA D, which is able to satisfy the request. DSA A then contains the merged data (which in this example came solely from DSA D), which it passes back to the requesting DUA.

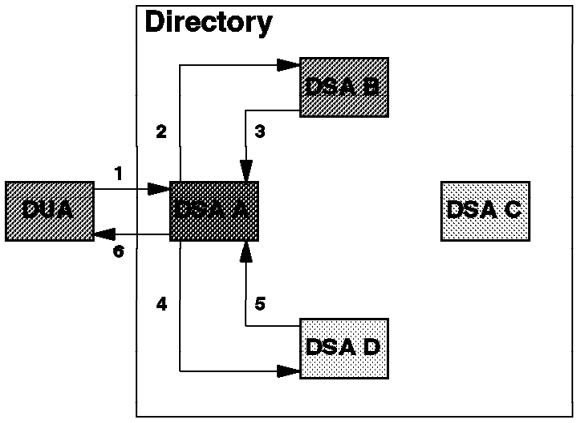


Figure 29. Referrals Pursued by DSA

Any combination of chaining and referrals is allowed by the standard. Which method is used to satisfy which requests depends on a variety of criteria: which

network connections are defined between DSAs, operating policies, security policies, and user preferences.

2.3.6.3 Knowledge Information

When a DSA receives a directory operation for a part of the DIT not contained locally, the DSA uses knowledge information to determine which other DSAs to pass the request to. This knowledge information is of various types, described in Figure 30.

Apart from some procedures to help with cross references, the 1988 standard does not define how this knowledge information is maintained by the various DSAs.

---

1. Superior Reference
   - Name of DSA holding superior entry
   - Presentation address of same DSA
2. Subordinate Reference:
   - RDN for the immediate subordinate entry
   - Name of DSA holding entry
   - Presentation address of same DSA
3. Non-Specific Subordinate Reference:
   - Name of DSA holding immediate subordinate entries
   - Presentation address of same DSA
4. Cross Reference (for Optimization)
   - The DN for an entry
   - Name of DSA holding that entry
   - Presentation address of same DSA

Figure 30. Knowledge References

2.3.6.4 Superior Reference

A DSA that administers any first level entry (immediately subordinate to the root: a country entry, for example), is known as a first level DSA. Any non-first level DSA must contain a superior reference to a DSA that is “closer” to a first level DSA. Following the chain of superior references eventually leads to a first level DSA. Note that while the standard specifies that a single superior reference be maintained, many implementations maintain several, usually one for each disconnected subtree, plus a default. This has no effect outside of the DSA, except to improve performance.

When a DSA receives a directory operation for a part of the DIT which is connected to its own portion of the DIT by entries closer to the root, the DSA uses the superior reference to identify the DSA to which all such requests could be sent. That DSA may forward them, in turn, using its own knowledge references.

Figure 31 on page 35 depicts Superior Reference.

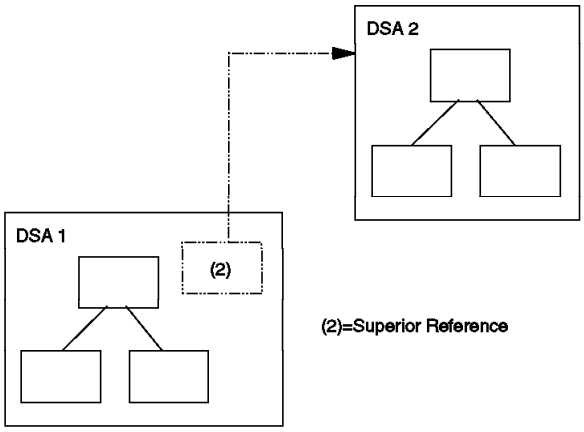


Figure 31. Superior Reference

2.3.6.5 Subordinate Reference

When a DSA receives a directory operation for a part of the DIT that is connected below an entry in its own DIT (that is, an entry in its own DIT has a DN which fully matches the leading portion of the DNs to be accessed), it uses subordinate references to decide which DSAs should receive the request. See Figure 32.

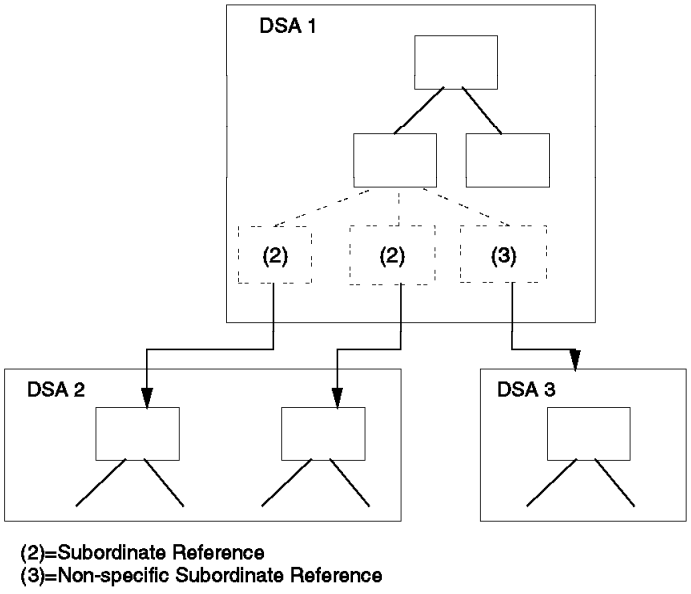


Figure 32. Subordinate Reference

A specific subordinate reference points to a specific entry at a DSA. A non-specific subordinate reference points to a DSA which may contain lower level entries, but what they are is not known by the higher level DSA.

2.3.6.6 Cross References

Cross references are used to short-cut the paths which might occur following superior and subordinate references through various DSAs. For example, a London DSA might have a superior reference to a UK DSA, which in turn has references to a US DSA. See Figure 33. Requests for entries in the US would be passed first to the UK DSA, which would then forward them to the US DSA. A cross reference in the London DSA to the US DSA would allow the request to be passed directly to the US DSA. Any number of cross references may be maintained, optimizing both superior and subordinate reference paths.

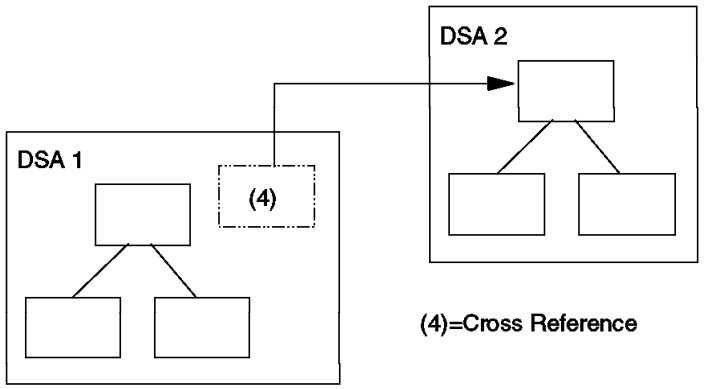


Figure 33. Cross References.

2.3.7 Service Controls

Normally, the user (or DUA on the user's behalf) will issue a request and let the DSA do the best it can. Because of the distributed nature of the directory, however, the user may wish to influence the way in which the request is handled. This is performed by the use of Service Controls. Some of these controls are requests, and may be ignored by the DSA, but others are orders and must be honored by the DSA.

2.3.7.1 Prefer Chaining

This control requests the DSA to use chaining, that is, to pass the request on to other DSAs rather than return a referral.

2.3.7.2 Chaining Prohibited

This control prohibits the DSA from passing requests on to other DSAs. This may be used where the DUA or application wishes to have control over where the request is passed, for security, operational, charging, or any other reason.

2.3.7.3 Local Scope

This control is similar to Chaining Prohibited, in that DSAs are prohibited from passing the request outside the "local scope." The definition of "local scope" is the responsibility of each implementation. It could be a single DSA or a group of commonly managed DSAs.

2.3.7.4 Don′t Use Copy

This control states that only master entries may be used to satisfy the request: no copies or cache entries are acceptable. This can be used to ensure an application has the latest information. Many applications can tolerate occasional "out of date" information, and trade off that cost against cheaper and faster access to more local copies.

2.3.7.5 Don′t Dereference Aliases

This control says that any alias name used in the request should not be resolved. This is normally used only to refer directly to an alias entry itself, when this control must be used.

2.3.7.6 Priority

May be set to low, medium, or high. How the directory service interprets this or how it modifies behavior is undefined.

2.3.7.7 Time Limit

May be set to any number of seconds. This is resolved in the initial DSA into an absolute time at which the request expires. If this time should occur before the operation is finished, DSAs stop further operation and return any information already obtained. Effectively, the initial DSA passes back the information it has available with an indication that the results may be incomplete. Any further results received are discarded by the DSA.

2.3.7.8 Size Limit

May be set to the maximum number of entries (not bytes) to be returned for List and Search operations. Because of the distributed nature of the directory, the set of results returned may be different for repeated requests. Any further results arriving at the initial DSA are discarded.

2.3.7.9 Scope of Referral

May be set to DMD or COUNTRY and limits the scope of referrals returned. For example, if a request were made to a UK DSA for a US entry, one could limit referrals to only those DSAs in the UK that knew about US, hence excluding referrals to DSAs that would require an expensive or insecure network connection. DMD (Directory Management Domain) limits the referrals to those DSAs managed by the same organization.

2.3.8 Search Guide

The 1988 standard defines (in X.520 ISO/IEC 9594-6) an interesting attribute — "Search Guide." This attribute provides information to help an application formulate a search request, by listing object classes and the attributes suitable for using in a filter when searching lower in the tree. For example, a country entry might specify a search guide for an organization object class, listing location, business category, and telephone area code as searchable attributes. The application could then build a panel requesting this information from the user. An organization entry might list "organization person" as a class with name, department number, and location as searchable attributes. Multiple search guides can be provided to cater to multiple object classes. The standard does not define how these should be used, but this facility could provide wider usable access into separately managed directory domains.

2.3.9 Security

The 1988 standard defines (in X.509 ISO/IEC 9594-8) an “Authentication Framework.” The standard notes that almost all security procedures rely on the reliable authentication of communicating partners; one partner must know that the other partner is who he claims to be. The directory is a natural place for authentication information to be placed. The authentication framework does not define the protocols used between applications to authenticate each other or users, but defines the types of objects (certificates, passwords, keys) and the techniques that may be used. The protocol for obtaining the information from the directory is the standard directory protocol.

You are recommended to read this portion of the standard and its annexes, as they provide a clear overview of the whole security area. You are also recommended to see the companion volume to this one, *The Library for Systems Solutions Security Reference.*

The 1988 standard provides no procedures or protocols for Access Control, that is, following authentication of a user, determining which information he may access. It contains, in Annex F to X.501, 9594-2, some principles which could be used by implementers. It contains information on the items requiring access control (subtree, entry, attribute, attribute value) and the categories of access (detect, compare, read, modify, add/delete, naming).

2.3.10 Areas Excluded from the 1988 Directory Standard

A number of functional areas are currently outside the Directory standard. Some may be included in later revisions, or be standardized by other OSI standards, but others are expected to remain outside the standard for the foreseeable future.

The following information referring to the 1992 standard was taken from the draft of that standard. Since this was a late draft, we are reasonably confident that this is what will appear in the final standard. It is still possible,however, that there will be some changes.

- Character set Specification

  The X.500 standard explicitly points to certain standards for attribute types, like country name and telephone number. For some other attribute types, this is left explicitly to application implementers and, hence, to groups like NADF (North American Directory Forum).

- NLS Implementation

  The manner in which DSAs in multiple countries with multiple national languages should cooperate in order to form a coherent distributed system is left to the implementers.

- Replication

  The manner by which copies of entries are created and maintained in DSAs, other than the master, is not defined by the 1988 standard. The 1992 standard provides standard replication.

- Access Control

  The manner by which protection is provided to the directory data is not defined by the 1988 standard. The 1992 standard provides standards for defining and exercising access control.

- Distributed Entries

  For some entries, it would be convenient for the administration and ownership of some attributes to be vested in a different DSA from other attributes. For example, the attributes of an organizational person may include some personnel related data that should be owned by the personnel DSA, and which may only be distributed to a limited subset of DSAs. However, there may be other attributes that are best administered elsewhere. X.500 1988 and 1992 defined a single master DSA being responsible for the maintenance of all attributes of an entry. While work was in progress to define this function for 1992, it was postponed to 1996.

  In X.500 1988 and 1992, other strategies must be used to satisfy this requirement. For example, a separate (subordinate) object may be defined just to contain the personnel information.

  In the case where multiple values for the same attribute are “owned” by different administrations, multiple entries can be made with different names. The some strategy is required to identify all the possible entries. NADF has recommended this approach for the situation where a customer is served by multiple Value Added Network providers (VANs) and hence has multiple addresses, each owned by a specific VAN. NADF has proposed a neutral name space with naming link attributes pointing to all the specific VAN entries.

- Schema Entries

  The manner in which a schema is communicated between systems is not defined in X.500 1988. X.500 1992 standardizes this by defining specific objects at specific access points in the tree. Normal directory operations are used to access this information.

- DUA API

  The manner in which a DUA and application communicate is not defined by the standard. This area is more likely to be standardized by industry interest groups, or by more general OSI standards for all application interfaces.

- End-User Interface

  The manner in which the directory requests are obtained from users and the results presented are not defined by the directory standard. This leaves application designers a free hand to integrate the directory function with any suitable application.

- Database Technology

  The manner in which the data is physically maintained is not defined by the standard. Any suitable technique may be used to provide the hierarchical tree-like logical structure. This is another area where creative thinking may provide a technical advantage. Typically, implementers have shied away from using relational techniques and are using basic or even purpose built access methods.

- Caching

  Having obtained information from a DSA during a directory operation, a DUA may keep that information and satisfy subsequent directory operations from that information, thus improving performance. The DUA must honor the “don′t use copy” service control element, and should have a satisfactory process to ensure the data is not too much out of date. Furthermore, the

DUA has to assure that the corresponding Access Control to the information is honored. The standard does not specify how a cache should be operated.

- Performance

  While the standard provides various controls and mechanisms for manipulating performance (size limit, priority, prohibit chaining, cross references), it does not state any performance targets. These are for agreement between interoperating agents.

- Data Content and Schema

  The directory may contain any data objects, attributes, and syntax. The scope of interoperability is limited to how far these objects, attributes, and syntaxes have been defined in the schema of other DSAs. The directory standard has defined a “starter set” and many applications will be built around this set, with or without extensions. Other interest groups and standards bodies will define other extensions of varying degrees of general interest.

2.3.11 NADF, GOSIP, EWOS, and Other Interest Groups

The standard is very rich and flexible, but also leaves a lot of areas undefined. Various organizations are now further refining the way in which the standard will work, by defining new object types or specifying meanings for the “locally defined” portions of the standard.

2.3.11.1 North American Directory Forum (NADF)

NADF is a collection of VAN service providers in North America who already interoperate electronic mail services and wish to interoperate directory services to support their mail services. NADF has defined the naming authorities and the schema structure to be used, and described a method to overcome the current limitation on distributed entries. IBM is a member of NADF and is represented by IBM Information Network as the service provider, and the Toronto development lab.

2.3.11.2 Government OSI Profiles (GOSIPs)

Several governments produce profiles stating what level of OSI support is required by vendors wishing to tender for government contracts. Since these contracts in total are large, these GOSIPs are influential in shaping products. The level of detail in a GOSIP varies considerably with the maturity and flexibility of a standard. Unfortunately, the GOSIPs (or their equivalent) from different governments differ in content and timing. Some work is in place to try to align the various GOSIPs.

2.3.11.3 Other Interest Groups

Many other groups in the open systems and standards arena are working on standardization and implementation profiles for directory. Efforts are being made to co-ordinate the activities to avoid duplication and fragmentation. Included in this list are the European Working Group on Open Systems (EWOS), the OSI Implementers Workshop (OIS), sponsored by the US National Institute of Standards (NIST) and IEEE, basically the US counterpart of EWOS and the Asian-Oceanic Workshop (AOW), the Asian counterpart of EWOS.

2.4 APPN Overview

This chapter gives a short introduction to Advanced Peer-to-Peer Networking*, describes the position of APPN in relation to LEN and SNA, and introduces the basic terminology used with APPN.

2.4.1 LEN and APPN

A network can be very simple, for example, two PS/2*s connected by a telephone line, as shown in Figure 34.

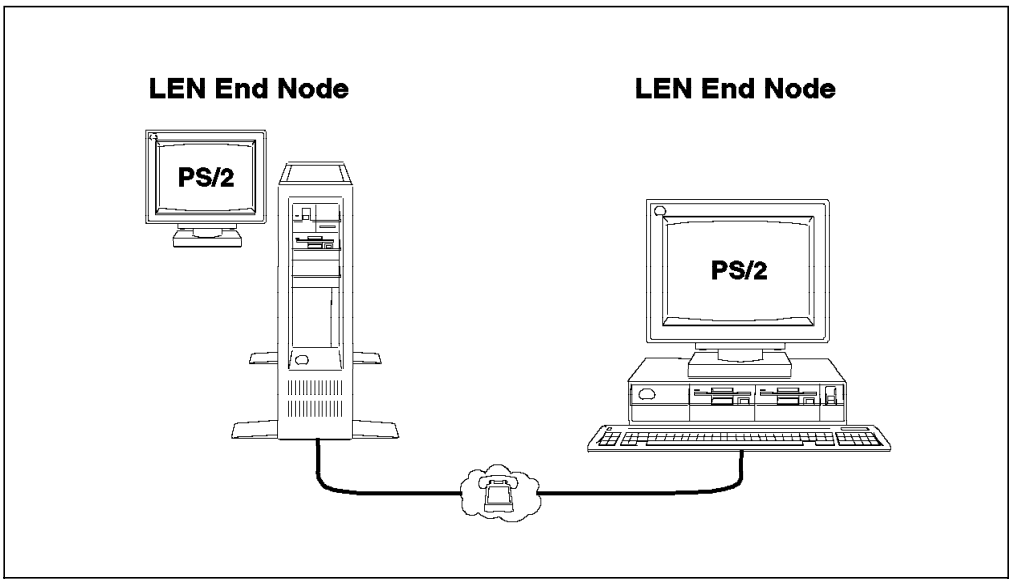


Figure 34. Two PS/2s Forming a LEN Connection

The purpose of connecting these two systems is to exchange data between two end users. An end user could be a person working with this system, a program running on the system, or a printer controlled by the system.

The end user gains access to the network through the logical unit (LU). Before the two LUs are able to exchange data, they must start an LU-LU session. For program-to-program communication, this session would typically be an LU 6.2 session.

In the case above, when the two systems (PS/2s) establish a low-entry networking (LEN) connection, the two connecting systems are known as ***LEN end nodes***. Using the architectural terms, the configuration above could be drawn as shown in Figure 35 on page 42. Several systems can be configured as LEN end nodes, such as VTAM* and NCP, AS/400* and PS/2. LEN end nodes provide the minimum functions required to:

- Provide a connection between LEN1 and LEN2
- Establish a session between the LUs named LUa and LUb
- Transport data.

The relation between LEN end nodes is truly peer-to-peer. Either side may activate a connection or start a session to the partner. It should be noted that according to the architecture, there are only *two* adjacent nodes involved in a LEN connection. That is, no matter how many nodes are actually in the network, the LEN connection only recognizes two nodes.

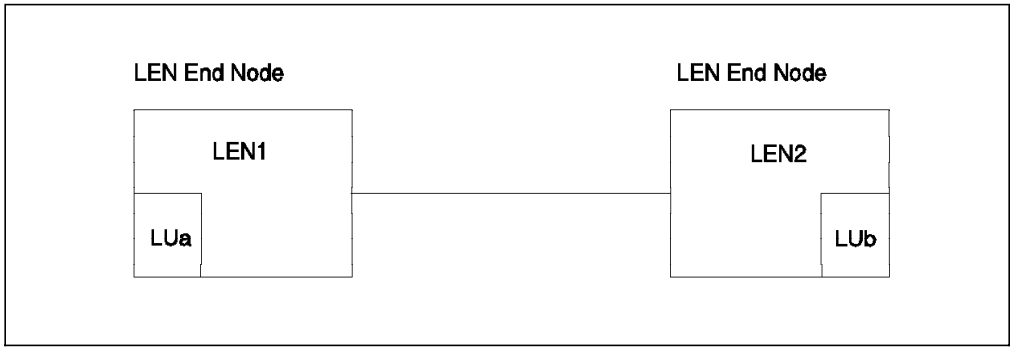


Figure 35. The Basic LEN Connection

Obviously, there must be functions in addition to LEN to build a network with more than two nodes. One of these functions is the capability to act as an intermediate node; that is, the node can receive data that is not for itself and can pass it on to the destination node. This principle is shown in Figure 36.

According to the LEN architecture, the relation between LEN end nodes is always a "*two*-node peer relationship." Therefore, LUs residing on nonadjacent LEN nodes can only establish sessions and exchange data, because the intermediate node presents itself as a LEN node owning all LUs residing on nonadjacent nodes. As seen from LEN1, the intermediate node is just a normal LEN end node, and LEN2 is not visible at all from LEN1. For LEN1, the LU named LUb seems to be in the intermediate node.

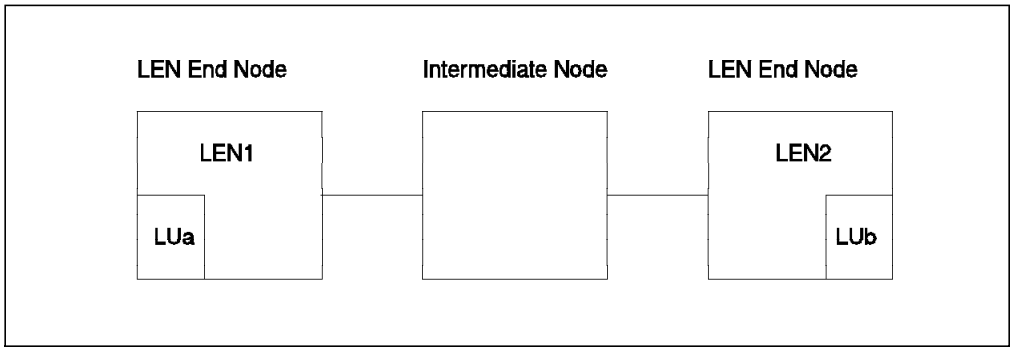


Figure 36. LEN End Nodes Connected to an Intermediate Node

VTAM and NCP support the LEN end node function and also provide intermediate routing between LEN end nodes. Figure 37 on page 43 gives an example of this configuration with VTAM on an ES/3090* as intermediate node.

The functions of LEN nodes are limited; for example, the nodes are not able to exchange topology and configuration data. Additional capabilities are required to reduce the number of definitions and the maintenance effort when building larger networks. For this purpose, the Advanced Peer-to-Peer Networking (APPN) architecture has been developed, which has been announced and published as the latest extension to SNA (Systems Network Architecture).

APPN architecture defines two basic node types:

- ***APPN End Node***

  The APPN end node is similar to a LEN end node, except that the control point (CP) of the end node exchanges information with the CP in the adjacent network node. The communication over the CP-CP sessions reduces the

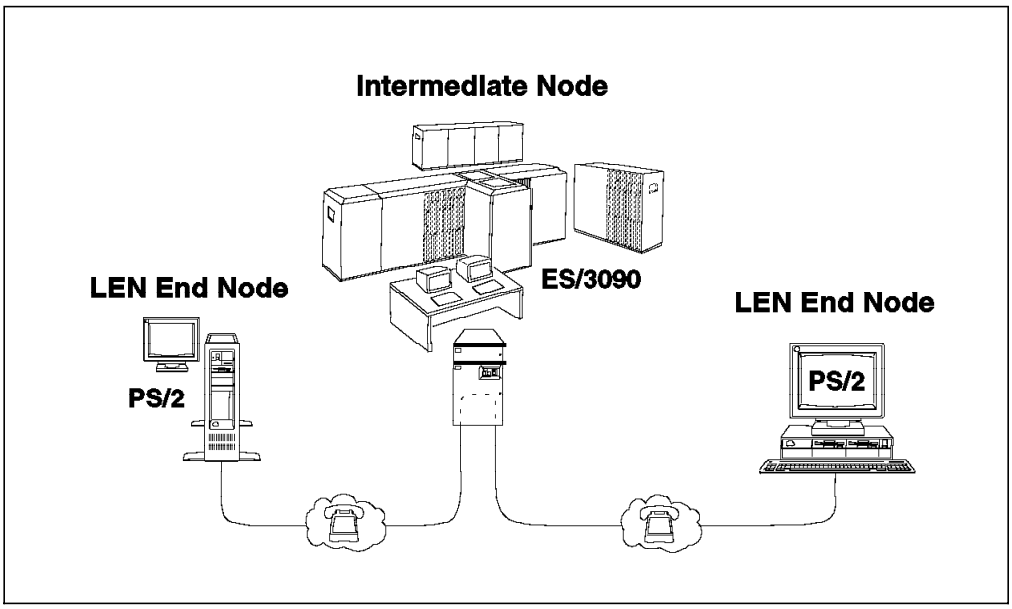


Figure 37. VTAM/NCP Providing the Intermediate Routing Function for LEN End Nodes

requirement for network definitions, and thus makes installation and maintenance of the network easier.

- ***APPN Network Node***

  The APPN network node has intermediate routing functions and provides network services to either APPN or LEN end nodes which attach to the network node. An APPN network node establishes CP-CP sessions with its adjacent APPN network nodes to exchange network topology and resource information. The CP-CP sessions to adjacent APPN end nodes are optional and are required only for APPN end nodes for which the APPN network node provides network services (such as locating resources in the APPN network).

APPN architecture describes the connection of LEN end nodes to APPN network nodes or APPN end nodes.

Figure 38 shows the basic form of an APPN network and gives an example of the services provided by the APPN network node. When LUa requests a session with LUc, the network node will locate the partner LU and assist in establishing the session.

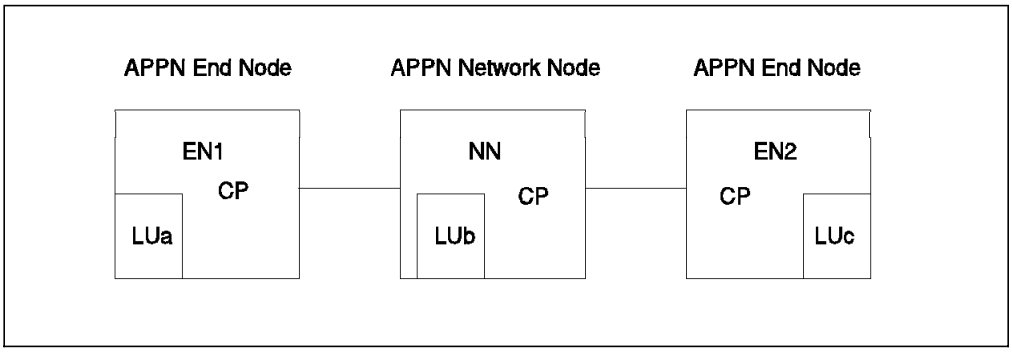


Figure 38. Advanced Peer-to-Peer Networking with Three Nodes

Figure 38 shows the basic form of an APPN network; however, APPN networks can be much more complex. APPN does not limit the number of nodes in an APPN network nor does it explicitly limit the number of intermediate APPN

network nodes through which LU-LU sessions are routed. One restriction exists however: the length of the Route Selection control vector (RSCV) describing a physical session path is limited to 255 bytes.

Figure 39 shows a backbone structure of APPN network nodes to which end nodes connect. The APPN nodes communicate through CP-CP sessions which are established between adjacent nodes. User sessions can be established from any LU to any LU.

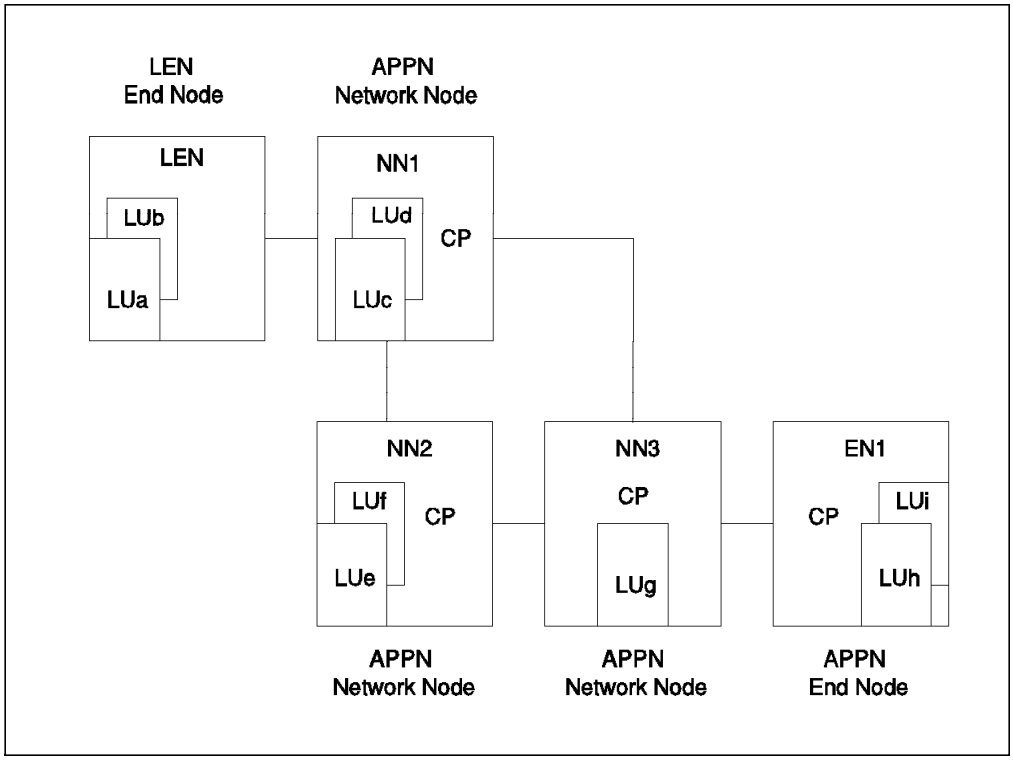


Figure 39. APPN Network with Different Node Types

While the previous figure showed the architectural node types used in the network, the next picture (Figure 40 on page 45) shows a variety of products, such as VTAM and NCP, AS/400, PS/2, and IBM 3174, connecting through different link protocols.

Figure 40 depicts a VTAM host, an AS/400, and an IBM 3174 configured as APPN network nodes, a PS/2 configured as an APPN end node and a second AS/400 configured as a LEN end node.

Note: A VTAM configured as a network node, together with all its owned NCPs, is called a *composite network node* (CNN). Within the composite network node, subarea protocols are used (see Figure 41 on page 45), but to the other APPN or LEN nodes, the CNN gives the appearance of an APPN network node.

You have seen that the APPN architecture defines several types of nodes and that the CPs of these nodes have different scopes of function. The internal implementation of the functions defined by the architecture may be different in different products.

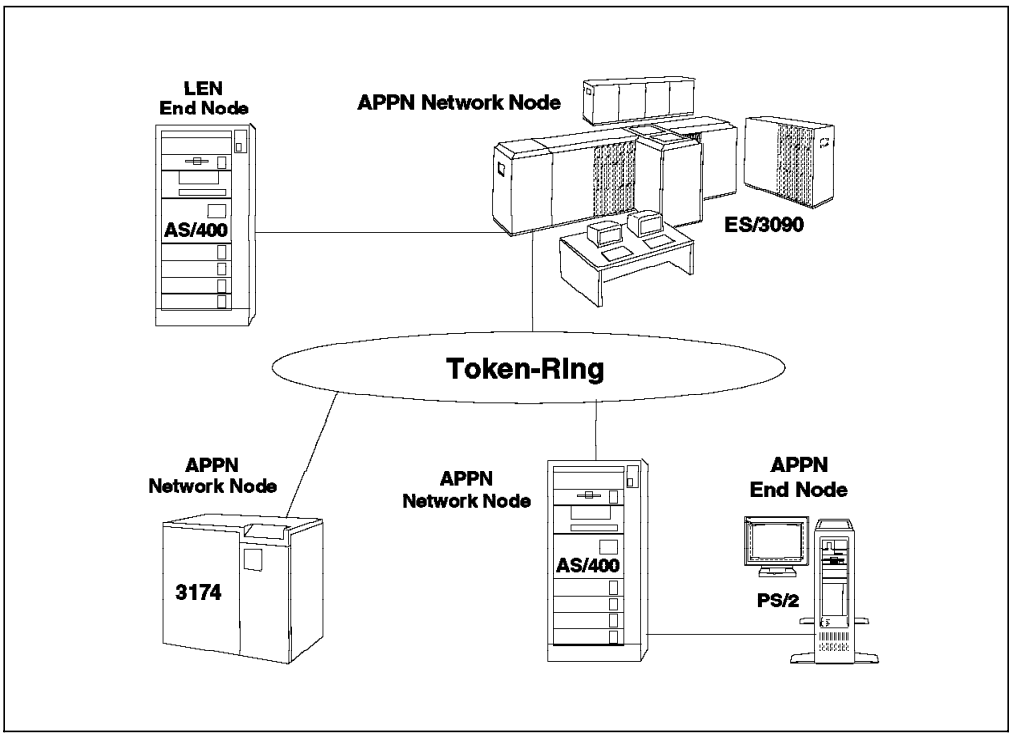


Figure 40. Advanced Peer-to-Peer Networking

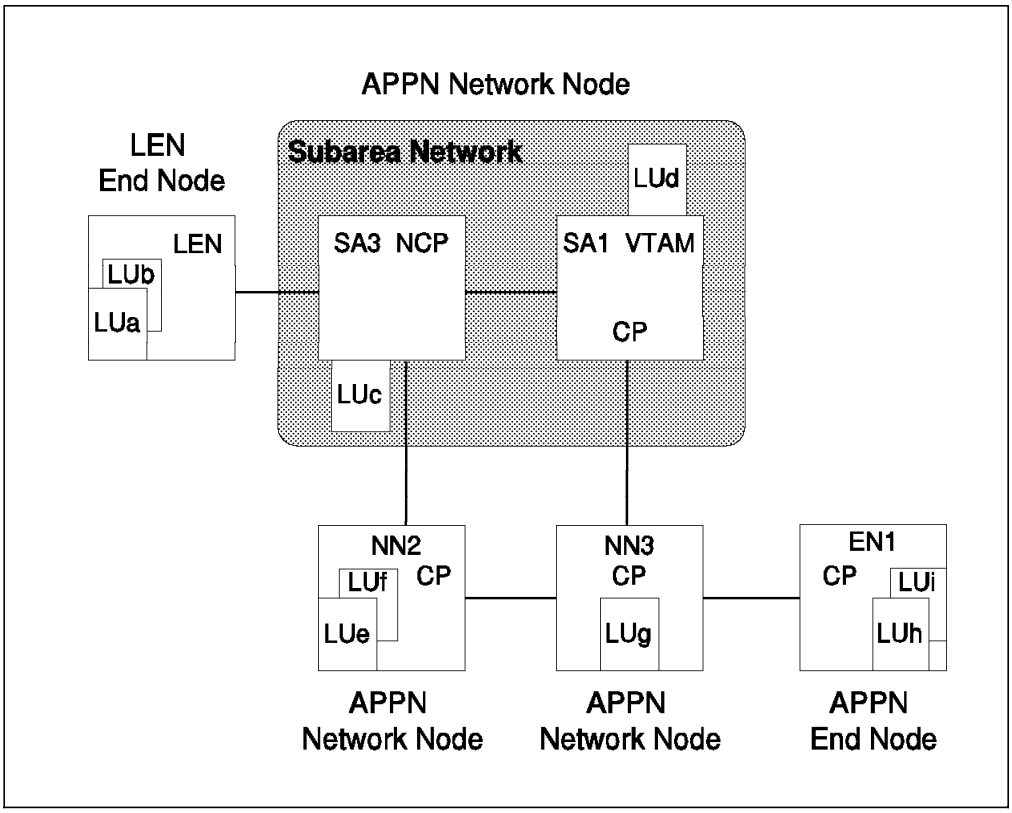


Figure 41. Composite Network Node with APPN Network Node Appearance

2.4.2 Names

Resource naming is important as it allows end users to start sessions without knowing the exact location of different resources within the network.

2.4.2.1 The Network Accessible Unit

In an APPN network, all components that can establish sessions with one another are called network accessible units (NAU). Examples are CPs and LUs. The term NAU was previously used as an abbreviation for “network *addressable* units.” The terminology has changed with APPN, and now NAUs are represented by names rather than by addresses.

Within an APPN network, the names of network accessible units must be unique. To ensure the uniqueness of names within the network, a consistent naming convention is a required. To make the administering of resource names easier, “the network” can be divided into partitions.

2.4.2.2 Network Identifiers

A partition of “the network” may be given a unique network identifier (net ID). Net IDs are 1 to 8 bytes long. The net ID is used throughout SNA, both in the subarea and the APPN part of a network. Because names of LUs and CPs have to be unique only within the scope of a net ID, they can be assigned and administered independently for each distinct partition of the network.

Registering can help network administrators ensure the uniqueness of the net ID they use. IBM* provides a worldwide registry for network IDs; information on the registration process can be obtained from your IBM representative.

IBM-registered net IDs should have an 8-byte name with the following structure:

cc is the country code (according to ISO Standard 3166).

eeee is the enterprise code (unique within a country).

nn is the network suffix code (unique within one enterprise).

2.4.2.3 Network Names

A network name is an identifier of a network resource. Each CP, LU, link, and link station in an SNA network has a network name. The network names are assigned through system definition. In an APPN node, the system definition is done using the node operator facility (NOF).

2.4.2.4 Network-Qualified Names

A network-qualified name identifies both the resource and the network in which the resource is located. It is a concatenation of the network ID and the network name of the resource; for example, the names NETA.LUA and NETB.LUA refer to different entities.

2.4.3 Addresses

Network addresses uniquely identify a resource throughout the subarea network. Local addresses uniquely identify a session on a link. APPN uses local addresses. Traditional SNA subarea networking uses local *and* network addresses. Local addresses are used between peripheral nodes and the boundary functions of VTAM and NCP; network addresses are used when routing data between subarea nodes and bear no relation to specific sessions.

Rather than being the address of the NAU, the “address” used in an APPN transmission header is an identifier unique on the given link for a particular session

Addresses are used for routing. Routing in an SNA network uses a combination of two things:

- Information carried in the transmission header of the message
- Information stored in the intermediate node.

In an APPN network, routing information is session oriented. The transmission header carries session identifiers that are locally defined for each pair of adjacent routing nodes and are only *temporarily* assigned. They are assigned at session initiation and released when the session ends. The session initiation request (BIND) carries routing information about the full session path that determines the sequence of links used from origin to destination. The local session identifier stored in each intermediate node in a session path is contained in a session connector and only kept for the life of the session.

The session identifier is associated with:

- A particular session
- A transmission group (link) between two nodes.

Figure 42 shows a session between two LUs, LUa and LUb, residing on two nonadjacent APPN end nodes. The session data is routed through two intermediate network nodes. The session can be thought of as a sequence of three session stages or, “hops,” with a distinct session identifier assigned to each session stage.

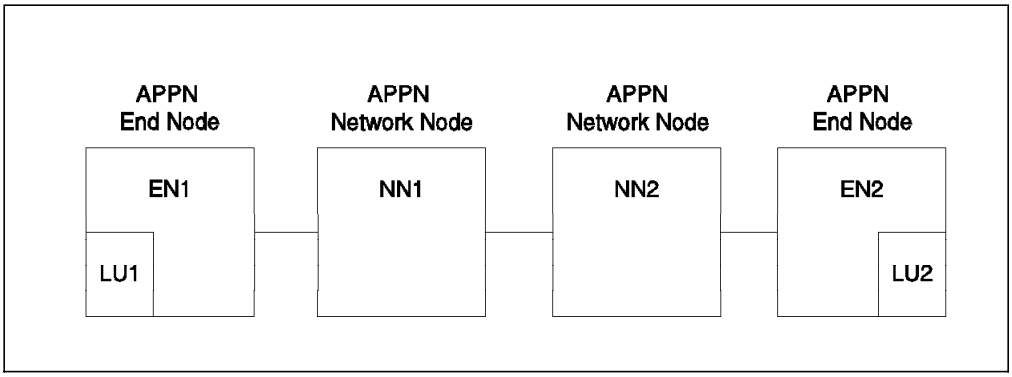


Figure 42. Session with Several Session Stages

Session identifiers vary at different session stages, which is why they are called local-form session identifiers (LFSID). The LFSID is set up during session establishment by the address space manager component of the CP and assigned for the “lifetime” of an LU-LU (or CP-CP) session.

Each session is uniquely identified by a network-unique identifier, the fully qualified procedure correlation ID (FQPCID), which is described in *APPN Architecture and Product Implementations Tutorial*.

In an HPR network, a new form of routing is used, which is called automatic network routing, or ANR. ANR is a source-routing protocol, which means the sender of a packet provides, in the network header, the information about the physical path the packet will use through the network. As HPR provides the

ability to do nondisruptive path switching, the HPR architecture handles the case where the route changes in mid-session.

ANR uses a new form of addressing to identify the route through an HPR network. However, unlike the APPN session-oriented addresses (LFSIDs), the addresses in ANR are based purely on the links which make up the route. The network header contains a list of ANR labels which identify the route through the network. Each ANR label describes a link which is to be taken to exit a node.

In addition to the ANR labels, there are still addresses that are associated with sessions in HPR. Each session will have a pair of unique session addresses, one for each direction. Unlike the LFSID, which identifies each stage of the APPN session, the HPR session addresses are used only on an (HPR) end-to-end basis. These are known as enhanced session addresses.

The process of supporting the end-to-end sessions across the HPR network is known as rapid-transport protocol or RTP.

In a network which is supporting both existing APPN nodes and HPR nodes, both the APPN and the HPR methods of addressing are used.

2.4.4 Domains

A domain is an “area of control.” A domain in an APPN network consists of the control point in a node and the resources controlled by the control point. Consequently, all APPN networks are multidomain networks.

Though all APPN nodes are peers with respect to session initiations and do not rely on other nodes to control their resources, APPN end nodes and LEN end nodes use the services of network nodes. The domain of an APPN end node or LEN end node contains the node′s own (local) resources. The domain of an APPN network node contains its local resources *and* the resources of those nodes that use the network node′s services. Thus, the domains of the APPN end nodes and LEN end nodes are included in the domains of their respective network node servers.

Note: In traditional subarea networking, a domain is the part of the network managed by a VTAM System Services Control Point (SSCP). Within this document, when using the term domain, we refer to an APPN domain unless explicitly stated otherwise.

2.4.5 Node Types

Before and after its announcement in 1986, the LEN end node was known by many names. Some of the names for the LEN end node that are found in various publications are:

LEN end node
LEN node
Peer node
PU type 2.1
PU 2.1
SNA PU 2.1
SNA Type 2.1 node
Type 2.1
T2.1
etc...

All the names mentioned above are synonyms for *LEN end node*. They all refer to the same function set. With the APPN extensions to SNA, two other types of nodes, *APPN end node* and *APPN network node*, have been introduced. Because VTAM, as an APPN node, identifies itself as a T5 node to the APPN network, it is no longer valid to use the term *T2.1 node* when referring to an APPN node. Throughout this document, the term *APPN or LEN node* will be used to refer to any of these three types of nodes, and the term ***APPN node*** when referring to either an APPN network node or an APPN end node.

2.4.5.1 APPN Network Node

An APPN network node provides distributed directory and routing services for all LUs that it controls. These LUs may be located on the APPN network node itself or on one of the adjacent LEN or APPN end nodes for which the APPN network node provides network node services. Jointly, with the other active APPN network nodes, an APPN network node is able to locate all destination LUs known in the network.

A facility known as ***central resource registration*** allows an APPN network node to register its resources at a central directory server. Once a resource is registered, APPN network nodes can locate the resource by querying the central directory server instead of using a broadcast search, thus improving network search performance during session establishment.

After the LU is located, the APPN network node is able to calculate the route between origin and destination LU according to the required class of service. All network nodes exchange information about the topology of the network. When two adjacent network nodes establish a connection, they exchange information about the network topology as they know it. In turn, each network node broadcasts this network topology information to other, active and adjacent, network nodes with which it has CP-CP sessions.

Alternatively, if the connection between network nodes is deactivated, then each network node broadcasts this change to all other, active and adjacent, network nodes. An APPN network node that is taken out of service will be declared inactive and, after some time, removed from the topology information in all network nodes together with its routing capabilities to other nodes.

The APPN network node is also capable of routing LU-LU sessions through its node from one adjacent node to another adjacent node. This function is called intermediate session routing.

2.4.5.2 APPN End Node

An APPN end node provides limited directory and routing services for LUs local to it. The APPN end node can select an adjacent APPN network node and request this network node to be its ***network node server***. If accepted by the network node, the APPN end node may register its local resources at the network node server. This allows the network node server to intercept Locate search requests for resources that are located on the APPN end node and pass the request on to the APPN end node for verification.

Without a network node server, an APPN end node can function as a LEN end node and establish LU-LU sessions with a partner LU in an adjacent APPN or LEN node.

The APPN end node sends Locate search requests to its network node server for resources unknown to the APPN end node. The APPN network node uses its distributed directory and routing facilities to locate the LU (via directed, central directory, or broadcast searches) and calculates the optimal route, starting at the &ae, toward the destination LU.

The APPN end node may have active connections to multiple adjacent network nodes; however, only one of these network nodes at a time can act as its network node server. The APPN end node selects its network node server by establishing CP-CP sessions with an adjacent APPN network node.

On APPN network nodes, the APPN end nodes are categorized as either ***authorized*** or ***unauthorized***. An authorized APPN end node may send registration requests to register local network accessible resources at a network node server, a facility known as ***end node resource registration***, and may, in addition, request that these resources be registered with the central directory server. If during session establishment a network node server does not know where an LU is located, the network node server queries authorized APPN end nodes within its domain which have indicated they are willing to be queried for unknown resources. Network accessible resources on unauthorized nodes require explicit definition at the network node server as part of its system definition or dynamically by the network node server′s operator. To avoid explicitly defining resources of authorized nodes at their network node server, the APPN end node should either register its resources or allow the network node server to query the APPN end node for unknown resources.

An APPN end node can attach to any LEN or APPN node regardless of its network ID.

2.4.5.3 LEN End Node

A LEN end node provides peer-to-peer connectivity to other LEN end nodes, APPN end nodes, or APPN network nodes. A LEN end node requires that all network accessible resources, either controlled by the LEN end node itself or on other nodes, be defined at the LEN end node. LUs on adjacent nodes need to be defined with the control point name of the adjacent node. LUs on nonadjacent nodes need to be defined with the control point name of an adjacent network node, as LEN end nodes assume that LUs are either local or reside on adjacent nodes.

Unlike APPN end nodes, the LEN end node cannot establish CP-CP sessions with an APPN network node; therefore, a LEN end node cannot register resources at a network node server, nor can it request its network node server to search for a resource, or to calculate the route between the LEN end node and the node containing a destination resource.

However, indirectly a LEN end node uses the distributed directory and routing services of an adjacent network node by predefining remote LUs, owned by nonadjacent nodes, with the CP name of an adjacent APPN network node. The session activation (BIND) request for that remote LU is sent by the LEN end node to the adjacent network node. The network node, in turn, automatically acts as the LEN end node′s network node server, locates the actual destination of the LU, calculates the route to it, and uses this route to send the BIND to its final destination.

A LEN end node can attach to any LEN or APPN node, regardless of its network ID.

2.4.5.4 Other Node Types

In SNA, a node represents an endpoint of a link or a junction common to two or more links in a network. The LEN end node, APPN end node, and APPN network node are endpoints of a link. Each node has a distinct role in an APPN network.

Besides these node types, you will find references in the APPN literature to other node types that are either synonyms for nodes as seen from a subarea network, represent a specific junction in the network, or represent an APPN node with additional functions. The following is not a complete list, but merely highlights the types of nodes found when creating this document:

- Boundary and peripheral node
- Composite node
- Interchange node
- Virtual routing node
- Peripheral border node
- Extended border node
- HPR node.

Boundary and Peripheral Node: Within traditional subarea SNA, the resources in a domain of a subarea SNA network are controlled through a hierarchical structure. The nodes that play a role in these networks are categorized as subarea and peripheral nodes. An example of such an SNA network is an S/390* type mainframe running VTAM and a 3745 communication controller running the Network Control Program (NCP). Both VTAM and NCP are referred to as subarea nodes. The VTAM subarea node includes the control point function, hereafter called the System Services Control Point (SSCP). Like the APPN control point, the SSCP controls all the resources that are in its domain.

Attached to these subarea nodes, or *boundary nodes*, are the *peripheral* nodes. The peripheral node is either a PU T2.0 or an APPN or LEN node. The PU T2.0 node is a traditional hierarchical node that requires the support of an SSCP to establish sessions. Traditional subarea SNA allowed LEN connections only; CP-CP sessions could not be established between VTAM and the APPN nodes.

With the introduction of APPN VTAM, a VTAM or a composite network node (subarea network consisting of one VTAM and one or more NCPs) is able to present an APPN image to other APPN nodes. APPN VTAM allows CP-CP sessions with APPN nodes attached to the VTAM or NCP boundary function, to get full APPN connectivity. The term “peripheral” node has lost its value in a network which is truly peer-to-peer.

Composite Node: The term *composite node* is used in some publications to represent a group of nodes that appear as **one** APPN or LEN node to other APPN or LEN nodes in an APPN network. For example, a subarea network that consists of one VTAM host and one or more NCPs consists of multiple nodes, but when connected to an APPN node, appears as **one** logical APPN or LEN node.

A subarea composite node may appear as either a LEN end node or as an APPN network node. In the former case, the term composite LEN node is used; in the latter case, the term composite network node (CNN) is used.

Interchange Node: A VTAM host acting as an interchange node (ICN) can be a stand-alone APPN VTAM node or a composite network node. The ICN routes sessions from APPN nodes into and through the subarea network using subarea routing, without exposing the subarea implementation to the APPN part of the network. This is accomplished by making the APPN VTAM node, plus all its owned resources, appear to other nodes as a single APPN network node with multiple connections. At the same time, the ICN, and the NCPs it owns, will maintain their subarea appearance to other subarea nodes.

The ICN supports SSCP-SSCP sessions with other VTAM nodes as well as CP-CP sessions with adjacent APPN network nodes and end nodes. This support allows the ICN to use both APPN and subarea data flows to locate LUs and to provide the best route between nodes. APPN session setup protocols, which flow on CP-CP sessions, are converted to the corresponding subarea protocols that flow on SSCP-SSCP sessions, and vice versa.

To an ICN, see, for example, VTAM1/NCP in Figure 43, multiple VTAMs and NCPs may connect using subarea protocols. Session establishment is possible between any LU in the subarea network and any LU in the APPN network. The VTAM host to which APPN nodes attach, or the VTAM host owning the NCPs to which APPN nodes attach, must have implemented APPN VTAM, as it is responsible (as an “interchange node”) for the conversion of subarea to APPN protocols and vice versa; other VTAMs within the subarea network may be backlevel VTAMs. From the viewpoint of the APPN nodes, LUs owned by VTAMs (for example, VTAM2 or VTAM3) other than the VTAM providing the interchange function are considered to reside on APPN end nodes.

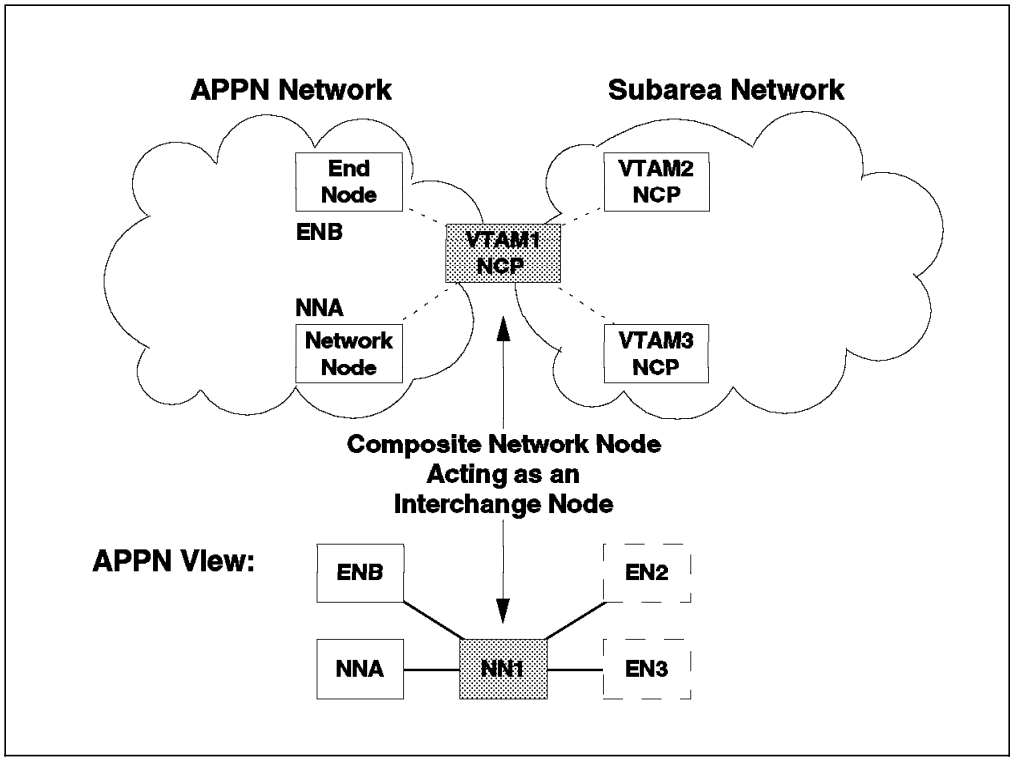


Figure 43. Composite Network Node Acting as an Interchange Node

Note: Figure 43 shows the basic form of connecting APPN and subarea networks using a composite network node acting as an interchange node.

Virtual Routing Node: APPN allows APPN nodes to reduce the addressing information stored at each node connected to a shared-access transmission facility (SATF), such as a token-ring, by allowing each node to define a virtual routing node (VRN) to represent its connection to the shared facility and all other nodes similarly configured. The SATF and the set of nodes having defined a connection to a common virtual routing node are said to comprise a ***connection network***.

A virtual routing node (VRN) is not a node, but it is a way to define an APPN node′s attachment to a shared-access transport facility. It reduces end node definition requirements by relying on the network node server to discover the common connection and supply necessary link-level signaling information as part of the regular Locate search process; LU-LU session data can then be routed directly, without intermediate node routing, between APPN nodes attached to the SATF.

Border Node: Base APPN architecture does not allow two adjacent APPN network nodes to connect and establish CP-CP sessions when they do not have the same net ID. The border node is an optional feature of an APPN network node that overcomes this restriction.

A border node can connect to an APPN network node with a different net ID, establish CP-CP sessions with it, and allow session establishment between LUs in different net ID ***subnetworks***. Topology information is not passed between the subnetworks. Similarly, a border node can also connect to another border node. Two types of border nodes are defined in the APPN architecture: peripheral border node and extended border node. (In the previous version of this book these were called Border Node Release 1 and Border Node Release 2, respectively.)

Peripheral Border Node: The peripheral border node enables the connection of network nodes with different net IDs and allows session establishment between LUs in different, adjacent, net ID subnetworks.

A peripheral border node provides directory, session setup and route selection services across the boundary between paired subnetworks with different network IDs while isolating each subnetwork from the other network′s topology information. This reduces the flow of topology updates and the storage requirements for the network topology database on network nodes in each of the network partitions.

Extended Border Node: The extended border node allows the connection of network nodes with different net IDs and session establishment between LUs in different net ID subnetworks that need not be adjacent.

An extended border node provides directory, session setup and route selection services across the boundary between paired or cascaded non-native net ID subnetworks. An extended border node can also partition a single net ID subnetwork into two or more clusters or topology subnetworks with the same net ID, thus isolating one from the topology of the other.

HPR Node: An HPR node is an APPN node that has implemented the optional HPR functions. An HPR node can be an APPN end node or an APPN network node.

In a mixed APPN and HPR topology subnetwork, a group of interconnected HPR nodes is sometimes referred to as an *HPR subnetwork*, or an HPR subnet. When an HPR link is activated between a pair of adjacent HPR nodes, an HPR subnet is formed.

In addition, the terms base APPN subnetwork and base APPN subnet may also be used when referring to a part of the network which is not an HPR subnet. HPR subnets are not separated from the other parts of the topology database.

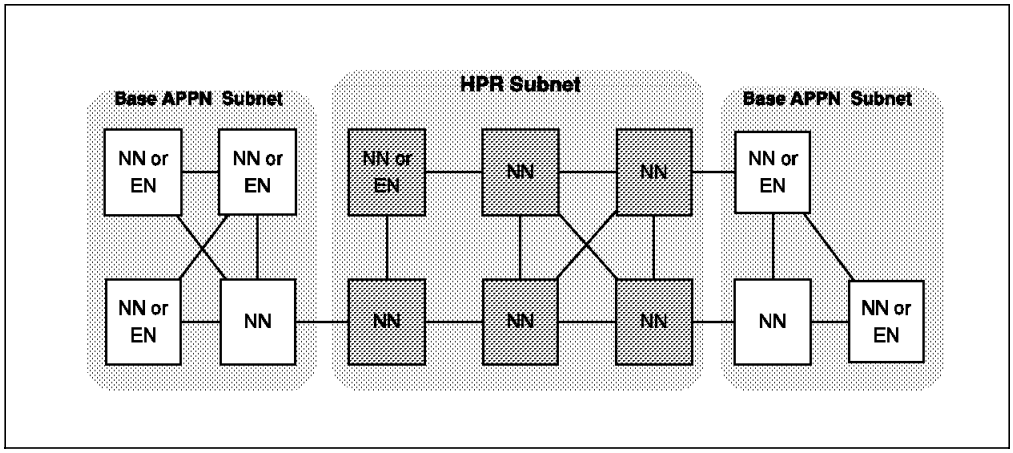


Figure 44. HPR Nodes and HPR Subnets

Figure 44 shows a backbone HPR subnet with two adjacent base APPN subnets. The six nodes in the HPR subnet are interconnected with HPR links.

If any of the nodes are to provide intermediate session routing, then they must be network nodes. But if a node acts only as a session end point, it can be a network node or an end node. The HPR nodes are exactly the same as APPN nodes in this respect.

If a product supports HPR, it can choose to implement only the base HPR function, or the HPR base function and optional functions. The base HPR function provides ANR routing, so as a minimum, an HPR node can always act as an intermediate node in an HPR network. An HPR node which is providing only ANR routing will always be a network node.

2.5 The IBM Open Blueprint

The IBM Open Blueprint is a structure for a distributed systems environment and provides the base upon which to build, execute, and manage distributed applications. The Open Blueprint builds on the Networking Blueprint and adds more detail to its upper layers. If you need more information about the Open Blueprint, please consult the list of documentation in the preface.

The Open Blueprint Directory provides a database of information about resources in the distributed system. Since resources in the distributed system follow standard naming conventions, passing these names to the directory services allows resource manager client functions to learn about the location of a resource and all information needed to interact with the responsible resource manager server.

The Open Blueprint Directory reduces the need for “side files” and other statically-defined configuration records. The directory enables a resource manager server, at the time a resource is created, to export information about that resource. Hence, client functions can dynamically learn the location of and how to access a resource, without its having been previously defined as part of the configuration.

The Open Blueprint Directory is based on the DCE directory technology from the Open Software Foundation. This technology provides truly distributed directory services. The contents of the directory can be spread across many systems. As a result, looking up the information associated with a name can involve interactions with several directory servers.

A name is passed to the appropriate directory server to begin the lookup process. This server inspects consecutive segments of the name hierarchy until the name has been fully parsed, in which case the directory returns the associated attribute values. If the directory finds a reference to another directory server, it passes this referral back to the requesting system, which then forwards the remainder of the name to the next specified directory server.

With this technique, although there is a single global name space, no single directory server has to contain all of the entries. Furthermore, because the name is parsed a segment at a time, successive segments can employ unique syntax, as defined for the server that will handle that name segment. This feature enables name formation by combining global names with cell names and even with resource manager-specific names.

The Open Blueprint provides two classes of general directory servers, as defined by DCE: the global directory service (GDS), an X.500-compliant directory service, and the cell directory service (CDS). Both directory implementations support distributed naming and server replication for backup or performance optimization.

CDS uses DCE RPC[3] for all client-to-server protocols (all name resolution services) and server-to-server protocols (such as replication). GDS uses communication protocols defined by OSI standards for both client-to-server and server-to-server protocols. In the Open Blueprint, both CDS and GDS protocols flow across multiple underlying transports using Common Transport Semantics. GDS provides a more robust set of name services, including the ability to look up unknown entries using attribute values. Both services allow public reads; however, specific authorization is required to update the directory.

A single programming interface, X/OPEN Directory Services/X/OPEN Object Manager (XDS/XOM), allows access to CDS and GDS. Using this API, an application is not concerned that a name may contain GDS and CDS parts.

Directory User Interface: The Open Blueprint includes a user interface for storing, retrieving, and searching information in the namespace. This graphical user interface provides capabilities for “browsing” information stored in GDS, CDS, or the security registry. In addition, it is an open structure that can be extended to allow resource manager-specific information to be interpreted and

[3] RPC: Remote Procedure Call. A means for communicating between two programs. Issuing an RPC is analogous to calling a subroutine, except that the subroutine may exist somewhere else within the network. RPCs are standard means of communication for the IBM Open Blueprint and OSF/DCE.

presented. Open Blueprint resource managers provide the extensions necessary to enable their unique directory information to be viewed and manipulated by users.

2.6 OSF/DCE Directory

As mentioned in the Introduction, the Open Software Foundation does not create new standards for its DCE components, but rather prefers to adopt "best-of-breed" existing standards. OSF adopted the DCE distributed naming services from DEC's DECdns** product and Siemens' DIR-X** X.500 services. As such, it represents a union of the two major directory architectures available, X.500 (for the global directory) and DNS (for the cell directories). The service discussed in this section is shown with the shaded box in Figure 45.

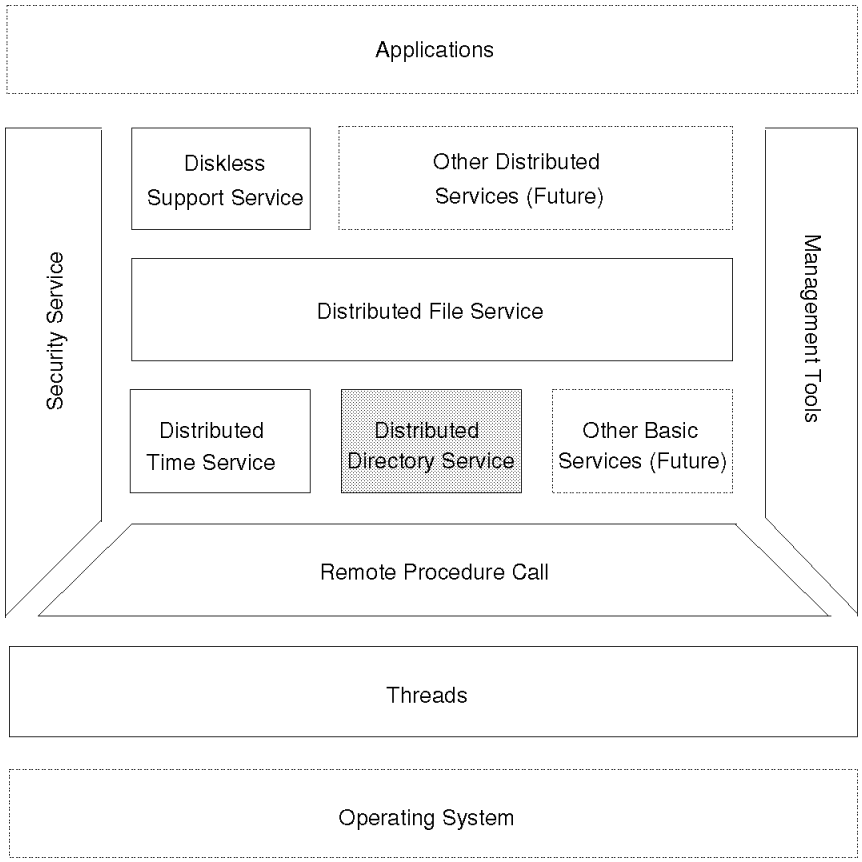


Figure 45. OSF DCE Directory Service

The Directory Service is the process that makes it possible for the user to locate objects in the network without knowing their physical location. It hides from the user the distributed nature of the environment. It is like a telephone directory assistance service that provides the phone number given a person's name. The users themselves – other than the DCE administrator – use the Directory Services indirectly, through an application interface. In this way, the application hides the use of the Directory Service from the user.

This section describes the concepts related to the Directory Services and the way it manages the naming environment in DCE.

2.6.1 Cell

DCE divides the distributed environment into administrative units (or domains) called *cells*. A DCE cell is a combination of client and server workstations.

A cell is a group of users, systems and resources that are typically centered around a common purpose and that share common DCE services. At a *minimum*, a DCE cell must have one cell directory and one security server. The number of systems and physical location does not define the cell boundaries but it is influenced by:

- Purpose - Different groups in an organization, like marketing or manufacturing or development, may want to share different resources and can determine the boundaries of a cell.
- Administration - There are a lot of administrative tasks related to DCE services and their respective databases. Depending on the availability of people, the organization can decide on distributing the DCE management to more than one group.
- Security - Security in a cell is a big task, and it can be interesting for the organization to define more cells so when there is a security break only passwords on that cell need to be changed. Security is covered in *Library for System Solutions, Security*, and will not be treated further in this document.
- Overhead - Usually there is more interaction within a cell so the boundaries have to be defined considering the kind of resources the users need to access and how often they access them.

Figure 46 gives us an example of a multi-cell environment.

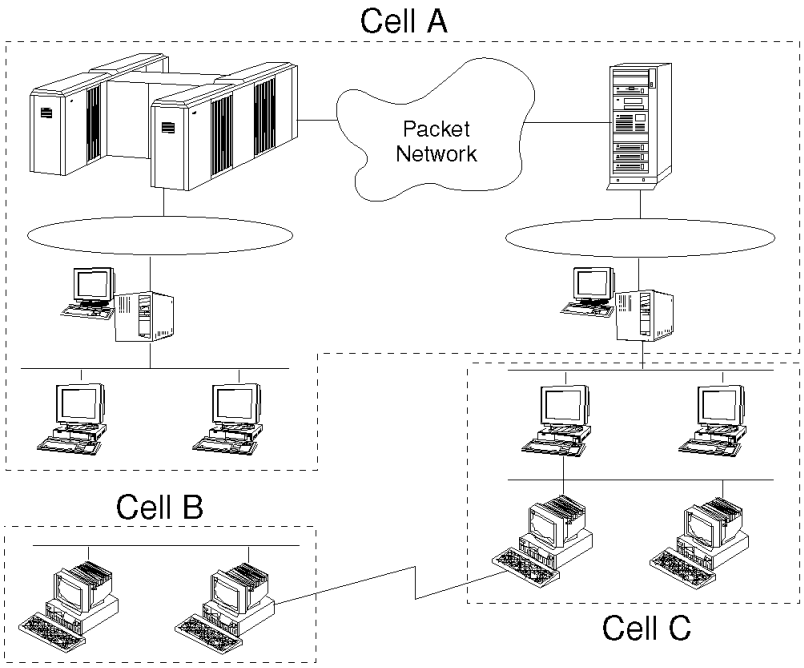


Figure 46. Multi-Cell Environment

As you can see from the illustration, cell A includes a very complex network including different LANs and WANs. Cell B is only a LAN and cell C includes two interconnected LANs.

2.6.2 Directory Services and the Cell

The Directory Service component that control names inside a cell is called the Cell Directory Service (CDS). The CDS stores names of resources in that cell so that given a name CDS returns the network address of the named resource.

Sometimes we need to find resources outside the cell. The Directory Service component that does that is called Global Directory Service (GDS) through an intermediate component called the Global Directory Agent (GDA). The various components of the Directory Services are shown in Figure 47.

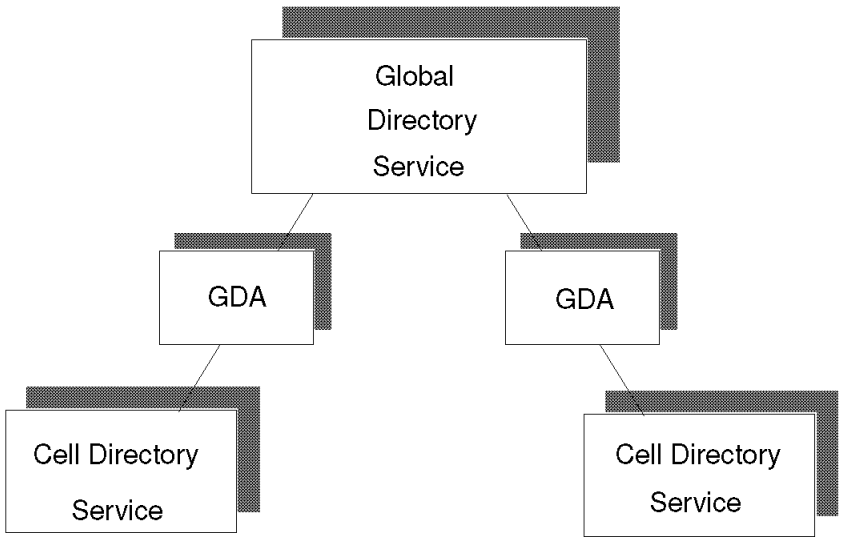


Figure 47. Components of the Directory Service

This is called a two-tier architecture.

The origin of the CDS is Digital Equipment's Distributed Naming Service (DECdns) and the origin of the GDS is Siemens' Dir-X implementation of the CCITT X.500/ISO 9594 international standard. X.500 is an emerging global directory service standard, but the Internet Domain Name System (DNS) is an established industry standard. For interoperability purposes GDS supports both, X.500 and DNS, transparently.

As seen in Figure 48 on page 59, when CDS determines that a name is not in its own cell, it passes the name to the GDA which can decide the global service used based on the name syntax.

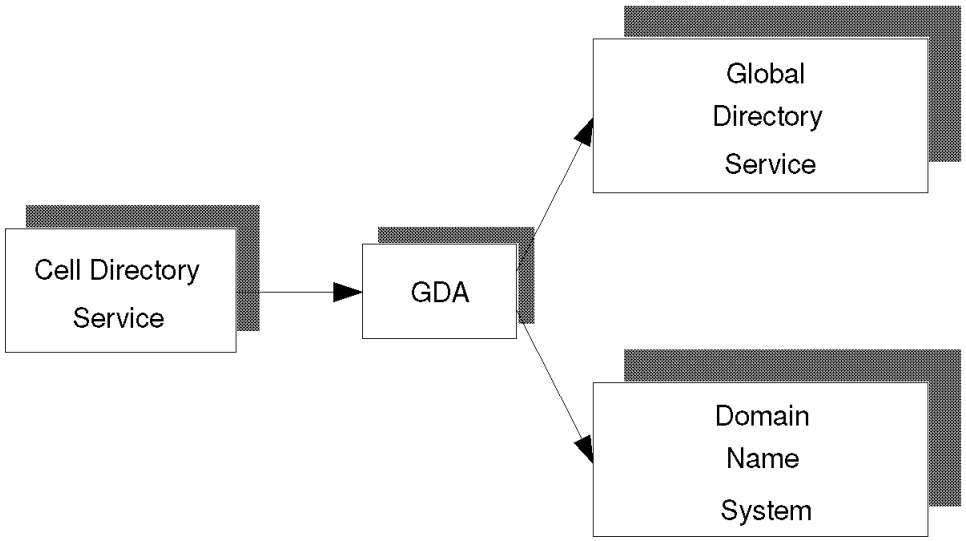


Figure 48. Interaction of CDS and GDA

2.6.3 Naming Environment

The DCE naming environment supports two kinds of names: global names and local names. The global name refers to names when outside a cell and local names when inside. Every object in the DCE Directory Service has a global name that is universally meaningful and usable anywhere in the DCE naming environment. The prefix (/...) indicates that the name is global. Following the prefix, the X.500 syntax defines four blocks, each one with two parts separated by an equal sign (**=**). The abbreviation of each block stands for country (C), organization (O) and organization unit (OU).

Figure 49 shows the name representing user Barbara at the ITSO cell in IBM.

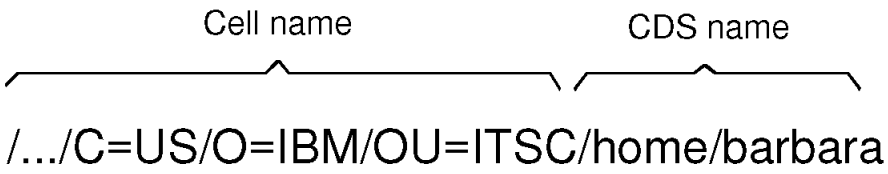


Figure 49. Representation of a User

Part of the name is the cell name and the rest is the CDS name. The cell name has to be unique to the global naming environment. The CDS name is the local name usable only within the cell where that entry exists. It is a shortened and more convenient form of a global name referring to resources within the cell.

The local name:

- Does not include a global cell name
- Begins with the prefix **/.:**

The prefix **/.:** indicates that the name being referred to is within the local cell, and when CDS encounters this prefix, it automatically replaces it with the local cell's name.

The following example shows two equally valid names within a cell:

- /.../C=US/O=IBM/OU=ITSC/home/barbara
- /.:/home/barbara

When referring to pathnames of files in the local cell, the name can be shorten by using the prefix **/:**. The default name of the file system root is **/.:**/fs, one level down from the root of the cell namespace. The prefix **/:** just substitutes the default name of the file system as seen in the following equivalent file names:

- /.../C=US/O=IBM/OU=ITSC/fs/users/barbara/herfile
- /.:/fs/users/barbara/herfile
- /:/users/barbara/herfile

2.6.3.1 CDS Names

The complete specification of a CDS name, going left to right from the cell root to the entry being named, is called the *full name*. Each element within a full name is separated by a slash (/) and is called a *simple name*.

CDS names can be organized into a hierarchical structure of directories that forms a tree structure. Multiple directory levels enable flexibility in distributing, controlling access to and managing many names. In this way, the following two names are unique names:

- /.:/home/barbara/sys/controlID/A948R21
- /.:/user/barbara/sys/controlID/A948R21

2.6.3.2 DNS Names

As defined earlier, the DCE naming environment supports DNS based on the Internet RFCs 1034 and 1035. The DNS is very common in many networks as a name service for hostnames. It can also be represented as a hierarchical tree with its topmost levels under the control of the Network Information Center (NIC).

The name directly under the root is a two letter code for country (such as *us* or *uk*) as defined in IOS standard 3166. Other names one level below the root include several generic administrative categories, such as *com* (commercial), *edu* (educational), *gov* (government) and *org* (other organizations). The owners of these names can grant permission to companies and organizations to create new subordinate names. Figure 50 on page 61 shows a comparison of the DNS hierarchical tree structure and the X.500 CDS representation.

X.500 picks the names in a top-down order while DNS does it in bottom-up order.

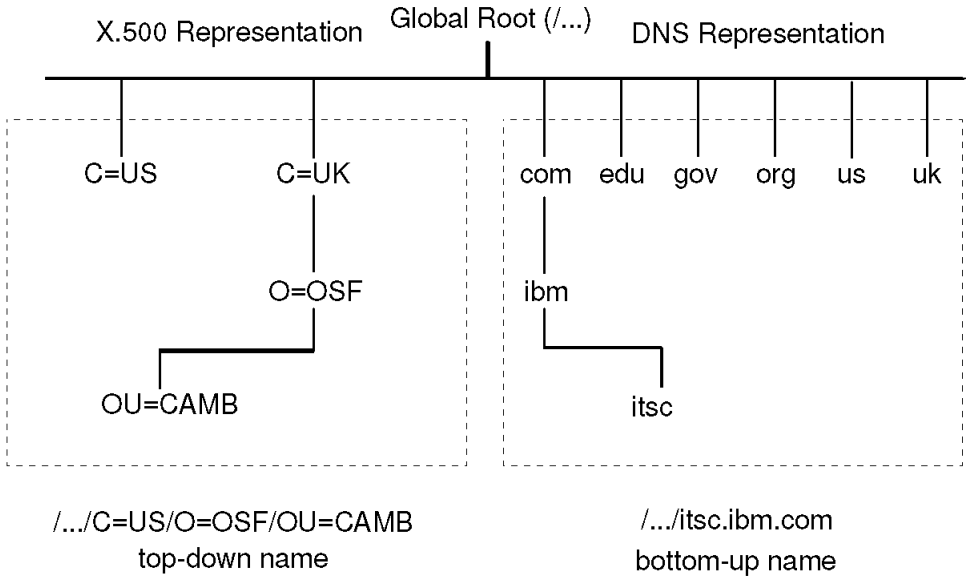


Figure 50. Comparison of Name Representation

2.6.3.3 DCE Namespace

The total collection of names shared by DCE is called the *DCE namespace*. It is a set of names contained in the DCE Directory Service and has the hierarchical structure. Within the DCE namespace, each entry is a *cell namespace* which represents each cell in the DCE environment. Each DCE cell must be named or registered in the DCE namespace to be accessible in the global naming environment. It can be registered in the GDS or DNS global namespace. Figure 51 shows a DCE namespace with cell namespaces in it. Each cell namespace has the same organization where security namespace and DFS namespaces are examples of directories in the cell namespace.

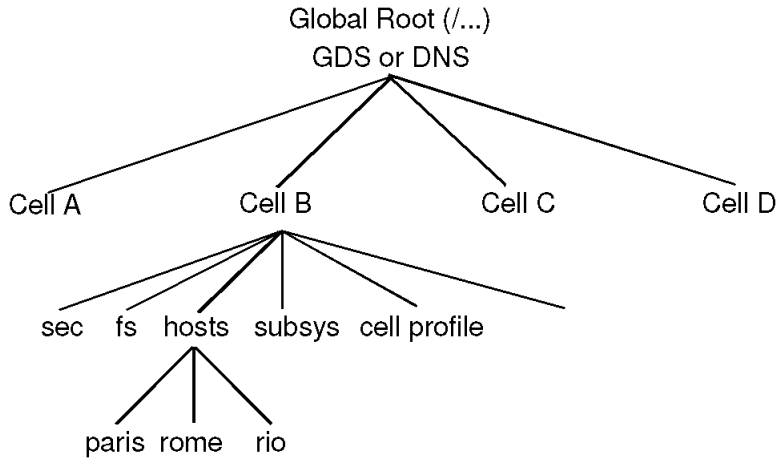


Figure 51. Example of a Cell Namespaces in a DCE Namespace

Not all DCE names are stored directly in the DCE Directory Service. As we have seen, some services, like the security server (*sec*) and the DFS (*fs*), connect into the namespace by means of specialized CDS entries called *junctions*. A junction entry contains binding information that enables a client to connect to a server outside of the Directory Service.

The security namespace is managed by the registry service of the DCE security component and has the following entries for security-related objects:

- Principals
- Groups
- Organizations.

The DFS namespace is managed by the file service of DFS with the entries being files and directories.

2.6.3.4 DCE Cell Namespace

Figure 52 shows the DCE cell namespace created by CDS for each cell in the DCE environment.

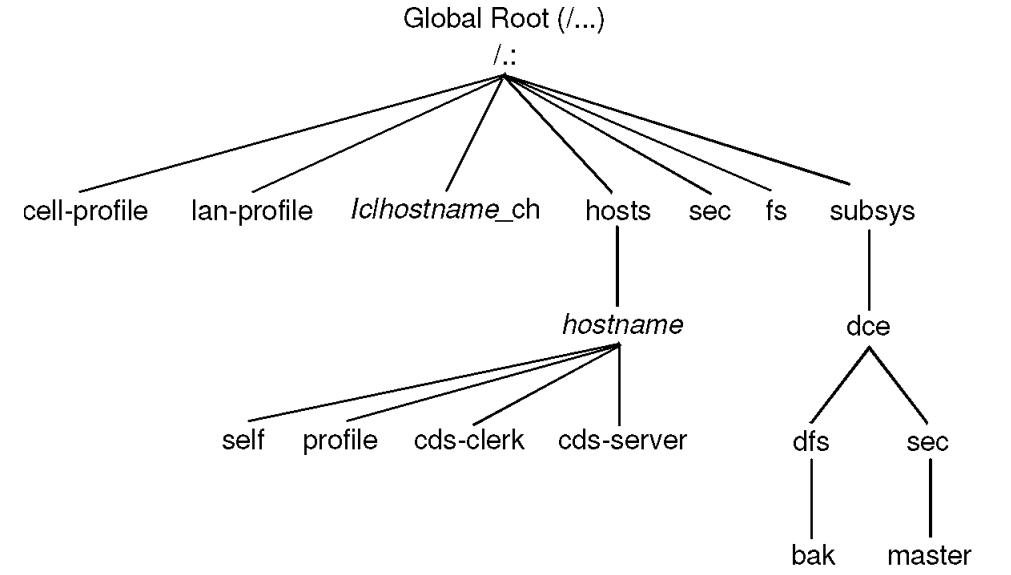


Figure 52. Directories Created for Each Cell

- **/.:/cell-profile** - this is the master default profile for the cell where all hosts, user and other profiles must chain up to
- **/.:/lan-profile** - this is the default LAN profile used by DTS and in single LAN cells, this is the profile in which entries for the time service local set entries are entered
- **/.:/***lclhostname*_**ch** - this is the clearinghouse of the cell
- **/.:/hosts** - this is where hosts directories are catalogued
- **/.:/hosts/***hostname* - each host has a directory in which RPC server entries, groups, and profiles associated with this host are stored
- **/.:/hosts***hostname***/self** - this entry contains a binding to the *rpcd* daemon on host *hostname*

- **/.:/hosts***hostname***/profile** - this is the default profile for host *hostname* and it must contain a default which points at /.:/cell-profile
- **/.:/hosts***hostname***/cds-clerk** - this entry contains the binding for a CDS clerk
- **/.:/hosts***hostname***/cds-server** - this entry contains the binding for a CDS server
- **/.:/sec** - this is the RPC group of all security servers for this cell
- **/.:/fs** - this is the RPC binding of all Fileset Database machines housing the FLDB
- **/.:/subsys** - this directory contains directories for different subsystems in this cell
- **/.:/subsys/dce** - this directory contains DCE specific names
- **/.:/subsys/dce/dfs** - this directory contains DCE specific names
- **/.:/subsys/dce/dfs/bak** - this entry contains the RPC bindings of all backup database machines storing the backup databases
- **/.:/subsys/dce/sec** - this entry contains security specific names
- **/.:/subsys/dce/sec/master** - this is the server entry for the master security server for this cell

2.6.4 Cell Directory Service (CDS)

The CDS manages to locate names within the cell and is optimized for local access. It is a partitioned distributed database service and the partitions can be stored in different locations allowing good scalability. The CDS can also be replicated allowing good availability of the system. A cache mechanism improves the performance by reducing the numbers of times it needs to be accessed. For the components of CDS, see Figure 53.

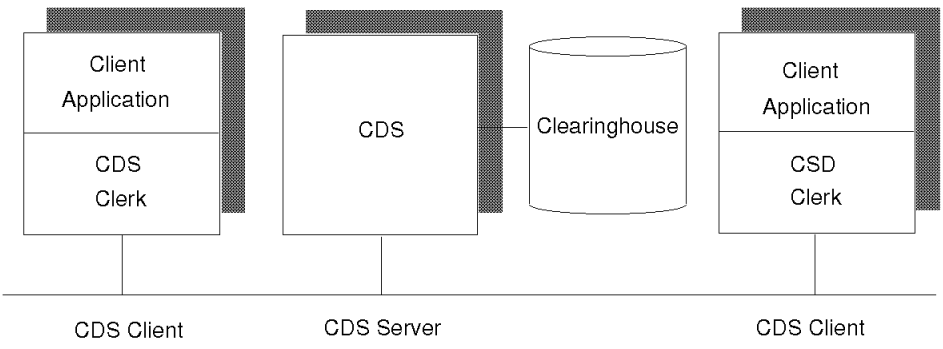


Figure 53. Components of a Cell Directory Service

Like any other DCE application, it follows the client/server model.

The CDS server manages a database of names and other information related to the entities of the cell, called a *clearinghouse*. The clearinghouse is where a CDS server adds, modifies, deletes and retrieves data on behalf of client applications. Each client runs a CDS clerk which intermediates the client applications and the CDS server. The clerk receives a request from the client application, sends the request to a server and returns the resulting information to the client. This process is called a *lookup*. The clerk caches the results of lookups so that it does not have to repeatedly go to a server for the same information. The cache is written to disk periodically so that the information can survive a system reboot or the restart of an application.

Figure 54 shows the lookup process:

1. The client application on node 1 sends a lookup request to the local clerk.
2. The clerk checks its cache and, not finding the name there, contacts the server on node 2.
3. The server checks to see if the name is in its clearinghouse.
4. The name exists in the clearinghouse, so the server gets the requested information.
5. The server returns the information to the clerk on node 1.
6. The clerk caches the information and passes the requested data to the client application.

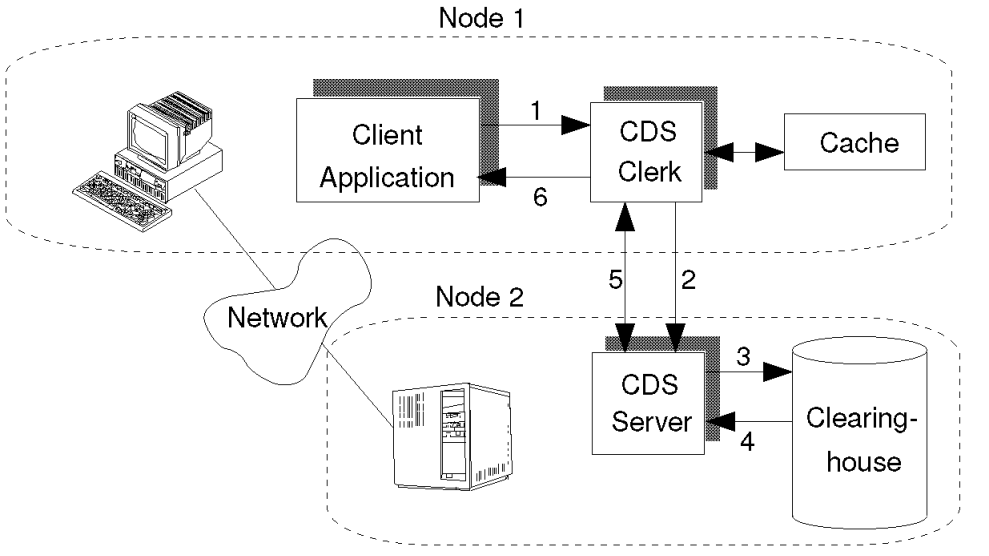


Figure 54. Sample CDS Lookup

2.6.4.1 Replicas

Directories are the units by which you distribute and replicate names throughout the namespace. Each physical copy of a directory, including the original, is called a *replica*. When you create a replica of a directory, you replicate all of the entries in it as well.

There are two types of replicas:

- Master
- Read-only.

The *master replica* is the first instance of a specific directory in the namespace. After making copies of the directory, it is possible to select another replica to be the master, but only one master can exist at a time. This master replica is the only directly modified replica of a directory.

The *read-only replica* is a copy of a directory that is available only for looking up information. CDS does not create, modify or delete names in read-only replicas. It simply updates them with changes made to the master replica.

Replicas can contain three kinds of entries:

- Object entries
- Soft links
- Child pointers.

Object Entries: When an object name is created, client applications and the CDS software supply attributes to be stored with the name. The name and its attributes make up the *object entry*. When a client application requests a lookup of the name, CDS returns the value of the attributes.

The *clearing house object entry* is predefined by the CDS and serves as a pointer to the location of a clearinghouse in the network. It is used by CDS to lookup and update data in a clearinghouse.

Soft Links: A *soft link* is a pointer that provides an alternate name for an object entry, directory or other soft links in the namespace. With the soft link, it is possible to do minor restructuring of a namespace or give multiple names for an object so that different users can refer to a name that makes more sense to them.

The soft link can be permanent or it can expire after a customized period of time.

A soft link has to be managed carefully because it is very difficult to keep track of those modifications on the namespace design.

Child Pointers: A *child pointer* connects a directory to another directory immediately beneath it in a cell namespace. Only CDS can create child pointers, and it is done automatically when a user creates a new directory.

The child pointer is created in the directory that is the parent of the directory to which it points. CDS uses child pointers to locate directory replicas when it is trying to find a name.

CDS Components Integration: The cell namespace consists of a complete set of names shared and managed by one or more CDS servers in a cell. The logical picture of a cell namespace is a hierarchical structure of directories and the names they contain. Every physical instance of a directory is called a replica. Names are physically stored in replicas, and replicas are stored in clearinghouses.

Figure 55 on page 66 shows the components of a CDS server node. Every CDS server manages at least one clearinghouse containing directory replicas. A replica can contain object entries, soft links and child pointers.

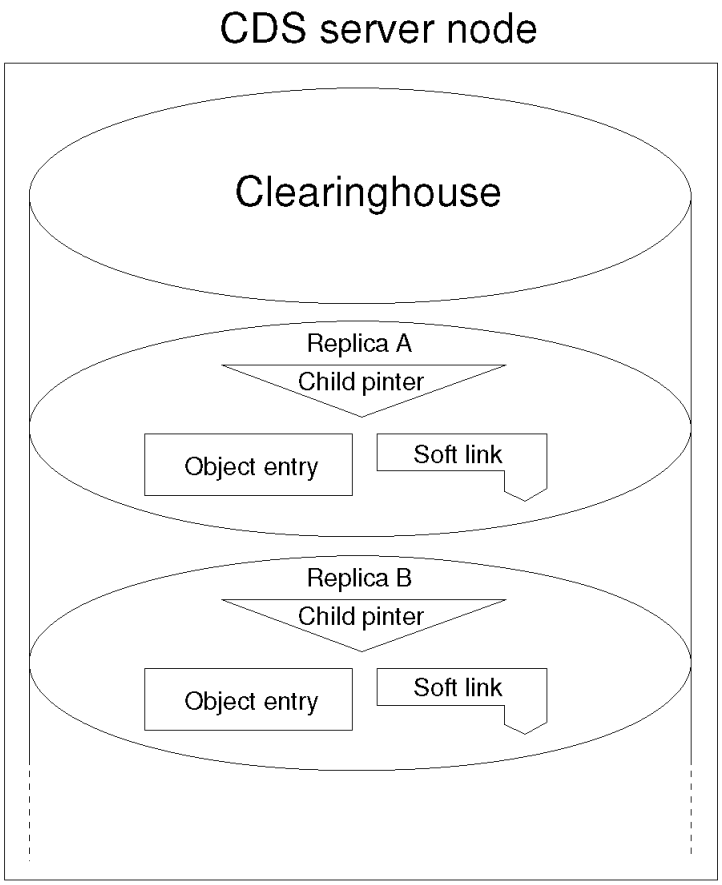


Figure 55. Components of a CDS Server Node

Figure 55 does not represent a real cell clearinghouse. For simplicity the figure shows just one directory in each replica and just two replicas in one clearinghouse.

2.6.4.2 CDS Lookup

CDS names are translated to physical resources used internally by CDS or by client applications. The attributes of a name are what make the translation possible.

Figure 56 on page 67 shows three directories and their contents in a logical namespace and how replicas of those directories are physically implemented in two clearinghouses.

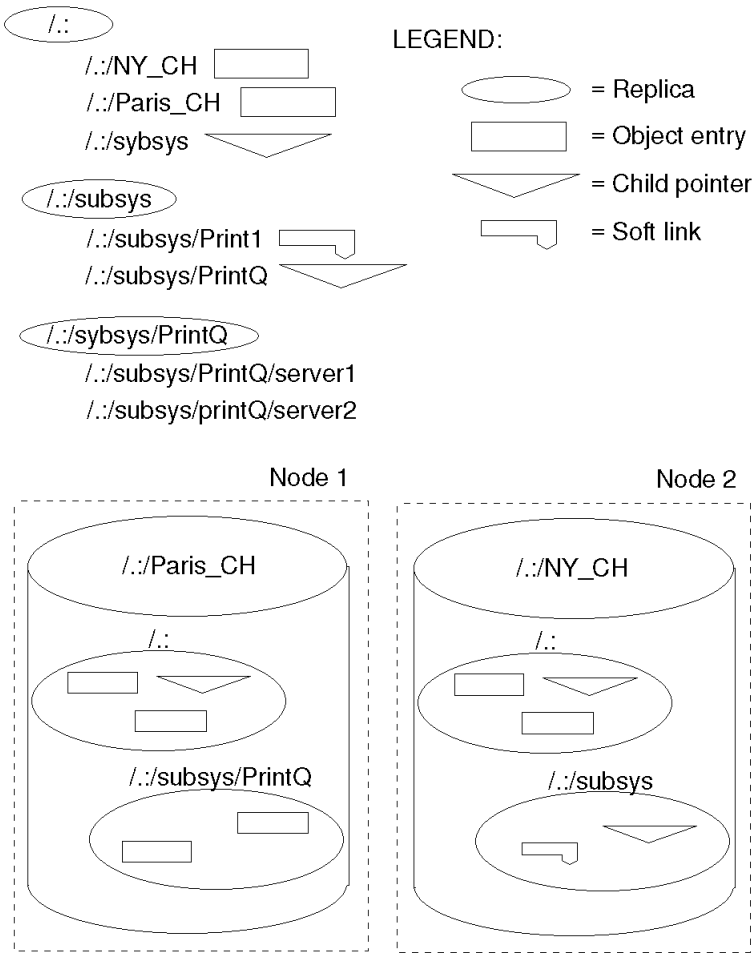


Figure 56. Logical and Physical View of a Namespace

The clearinghouses have CDS names: /.:/Paris_CH on node 1 and /.:/NY_CH on node 2. The _CH suffix is a recommended convention for naming clearinghouses.

/.:/Paris_CH clearinghouse contains replicas of the root directory and the /.:/subsys/PrintQ directory. /.:/NY_CH clearinghouse contains replicas of the root directory and the /.:/subsys directory.

Figure 57 on page 68 shows the relationship between two clearinghouse object entries and the clearinghouse they describe. The clearinghouse object entry is the only one created, used and maintained by the CDS. However, it is like any other object entry in that it describes a physical resource in the network: the clearinghouse. It is automatically created when you create and name the clearinghouse.

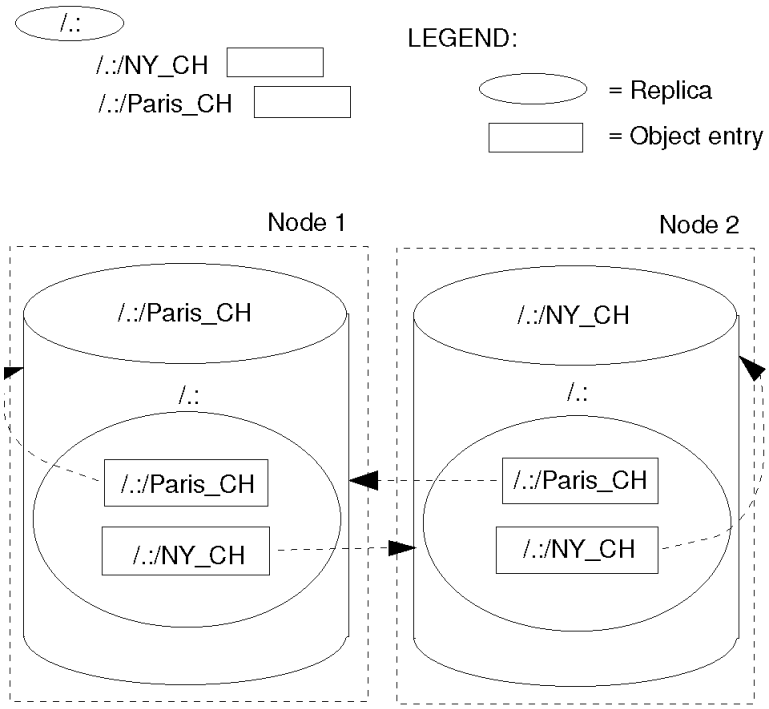


Figure 57. Clearinghouse Object Entries and Clearinghouses

Figure 57 shows two clearinghouses object entries: /.:/Paris_CH which points to the clearinghouse named /.:/Paris_CH on node 1 and /.:/NY_CH which points to the clearinghouse named /.:/NY_CH on node 2.

Figure 58 on page 69 shows the relationship between a soft link, the object entry it points to, and the resource that the object entry describes.

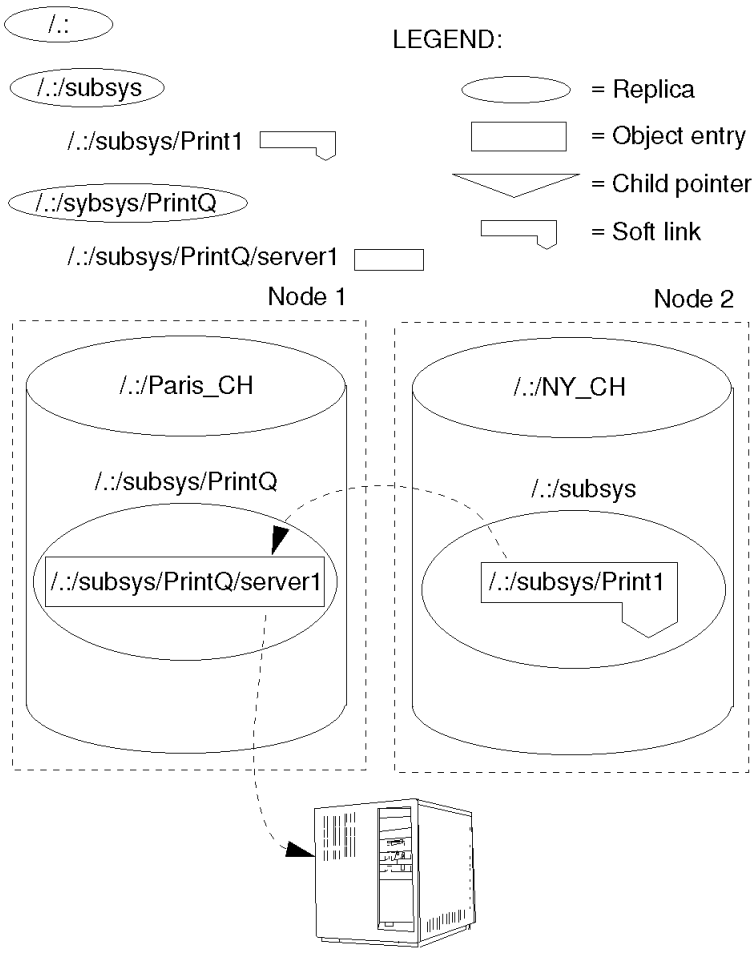


Figure 58. Soft Link and its Resolution

The soft link /.:/subsys/Print1 has an attribute called CDS_LinkTarget which contains the name that the link points to: an object entry named /.:/subsys/PrintQ/server1. The object entry describes a print server machine used by an application called PrintQ. The replica containing the /.:/subsys/PrintQ/server1 object entry exists in the /.:/Paris_CH clearinghouse. The object entry has an attribute called CDS_Towers, which contains RPC binding information that enables the PrintQ application to contact the print server machine.

Figure 59 on page 70 shows the relationship between directories and their associated child pointers. It illustrates that, although a child pointer has the same name as its associated directory, the child pointer is a separate entry in the namespace and resides in the parent of the directory to which it refers.

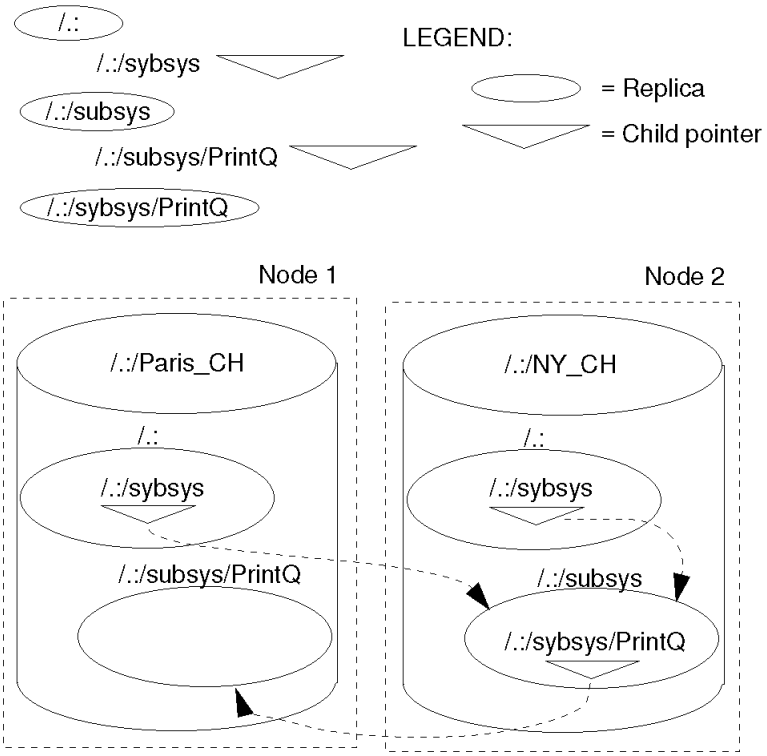


Figure 59. Child Pointers and Directories

The root replicas in both clearinghouses contain a child pointer for the /.:/subsys directory. The /.:/subsys child pointer has an attribute called CDS_Replicas which contains the name and address of the /.:/NY_CH clearinghouse, where a replica of the /.:/subsys directory exists. In the /.:/NY_CH clearinghouse, the replica of the /.:/subsys directory contains a child pointer for the /.:/subsys/PrintQ directory. The child pointer's CDS_Replicas attribute contains the name and address of the /.:/Paris_CH clearinghouse, where a replica of the /.:/subsys/PrintQ directory exists.

2.6.4.3 CDS Clerk

Before being able to request a lookup to the CDS server, the clerk has to learn how to locate one CDS server in the DCE environment. There are three ways for the clerk to locate them:

- Solicitation and advertisement protocol
- During a lookup
- Management command.

The solicitation and advertisement protocol is used for the CDS server to broadcast messages at regular intervals to advertise its existence to clerks in a LAN. The advertisement message contains information, like the cell that the server belongs to, the server's network address, and the clearinghouses it manages. At startup, the clerk sends out solicitation messages to request for advertisement.

During a lookup, if the CDS server does not find the name in its clearinghouse, it gives the clerk as much data as it can about where else to search for the name.

For example, if its clearinghouse contains replicas that are part of the full name being looked up, it returns data from a relevant child pointer in the replica it does have. The child pointer's CDS_Replicas attribute contains information of clearinghouse names and binding information.

The CDS administrator can issue the `define cached server` command to create knowledge in the clerk's cache about a server. This command is useful when the server and the clerk are separated by a wide area network (WAN) where the advertisement protocol does not work.

Chapter 3. Naming Technologies and Standards

As mentioned in the introduction, the increase in the interconnection of computer systems has led to the requirement for a global naming system that provides unique names for all manner of data processing entities. Building such a system and making it manageable represents a significant challenge.

Local naming is familiar to any user of a traditional computing system and will always be important because much computer activity will continue to take place within limited scopes. Names must be unique within those limitations, but there is no requirement that they be universally unique. This makes their administration much easier.

Most local naming schemes are integrated into the system that they support.

This chapter will divide global naming technologies into two groups:

- Technologies standardized by international standards organizations, such as CCITT and ISO/IEC. A special case here is the IBM Open Blueprint which emphasizes a subset of those technologies.
- Technologies that have been standardized by industry groups. Here the primary body is the Open Software Foundation, OSF.

Further, there are *de facto* standards, which were not instituted by international standards or industry bodies, but were developed to solve a specific problem and were then adopted by a wider circle of users. The primary *de facto* standard is the Internet Domain Name Service, which is described in the previous chapter.

3.1 Global Naming Standards

As the degree of interconnectivity becomes progressively greater, the need for global naming becomes ever more important. This is true not only for networked computer systems that exchange data objects, but also for aspects of human interaction, such as telephone calls and credit cards. The success of cellular telephones has generated new telephone numbers faster than expected, requiring carriers to increase the range of possible numbers— for example, by creating new area codes. If new developments in technology make personal, portable communications devices practical, the demand for globally unique telephone numbers will escalate even faster. Increasing use of credit cards also requires commercial institutions to develop robust identification systems that can insure uniqueness of numbers and efficient access to them.

A concept that has become increasingly used in conjunction with global naming is that of *namespace*. This reflects a growing awareness of the importance of naming and a realization that names are assigned with a scope. Names unique within a specific scope are adequate as long as all name-dependent activity is confined within that scope. In the case of the telephone, an example of a limited namespace would be the set of all extensions defined within the telephone at a company branch office. An instance of a name in this namespace might be a five-digit number. When telephone communication must take place in a larger scope, however, limited namespaces collide and more sophisticated names (telephone numbers) become necessary. Thus, placing a call from an overseas

location to a person at the local branch office invokes a naming system at a higher level operating within a much larger namespace— a global namespace.

It is obvious that naming in a global namespace cannot be administered by a single user of that namespace (unless the user happens to have a monopoly on the service involved). It is therefore necessary to have standards that are universally accepted and have their highest levels administered from a central authority.

The international standards community is actively developing naming systems to support global communication. This activity has resulted in publication of several ISO and CCITT standards relating to global naming. At the national level standards organizations, such as ANSI, are gradually recognizing the need for effective national registration authorities, which are necessary for implementation of unambiguous global naming. Even though technical development of naming systems is not sufficient in itself and rigorous administrative procedures must be established to govern the issuance of names, this document will concentrate on the technical aspects of naming and only mention the administrative aspects where that contributes to the technical understanding.

Additionally, this chapter discusses the naming technology specified in the IBM Open Blueprint for open heterogeneous systems. These standards are subsets of existing international standards.

3.2 Naming in the IBM Open Blueprint

A major goal of the Open Blueprint is to achieve a seamless, single-system image across a heterogeneous collection of systems. To accomplish this, a consistent approach to naming must be established across all resources of the distributed system.

Today′s operating systems frequently define one or more namespaces unique to the system. These namespaces may be defined by operating system convention or shaped by specific resource managers. In an open, heterogeneous, distributed environment, the potentially large numbers of resource types and implementations can create a complex array of naming conventions, with unique syntax and approaches to context.

3.2.1 Universal Naming

To simplify the tasks of using, administering, and writing applications in an open, heterogeneous, distributed environment, the Open Blueprint includes a universal namespace based on the structure implemented in OSF′s DCE technology. With universal naming, resources of any class, such as programs, hardware, data, and users in any location, can be referred to by a name that follows a single set of naming rules.

The universal namespace includes two distinct sub-namespaces: the global namespace and the local, or cell, namespace. Global names use the ISO X.500 or Internet naming standards. Cell names follow the Cell Directory Service (CDS) naming conventions (untyped) as defined by DCE. Resource names can be added in either namespace. Cell names refer only to resources within a single cell. To access resources in a “foreign” cell, the global name of the cell

is combined with the cell-relative name. Figure 60 on page 75 shows an example of each of these universal naming formats.

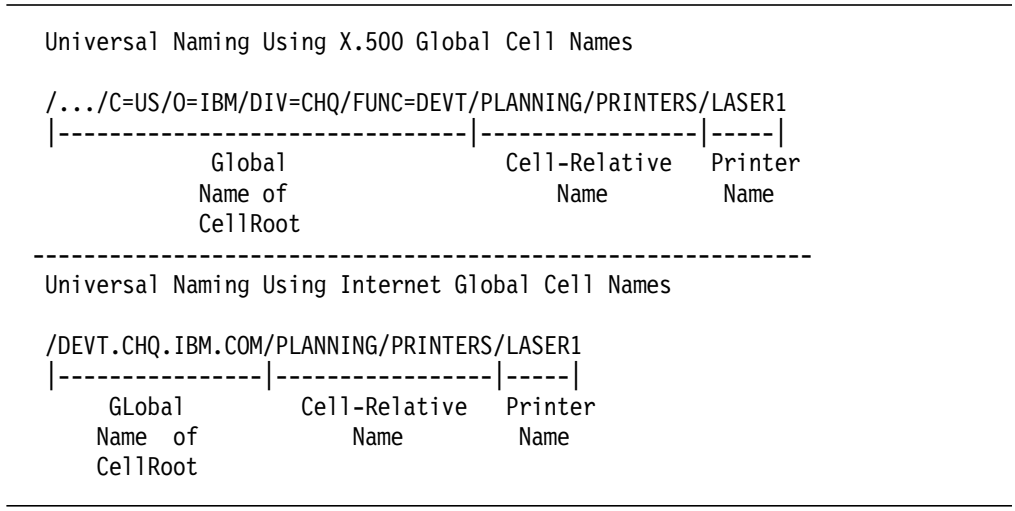


Figure 60. Universal Naming Examples

Over time, new resource managers, APIs, user interfaces, tools, and application programs will support the universal name space. Existing system and resource manager-defined namespaces must, of course, continue to be supported. Resources may be referenced either by their universal names or through a resource manager-defined name.

3.2.2 Concatenated Naming

In addition to the disparate syntax and structure of today's resource manager name spaces, each is defined within a name (and usually administrative) scope established by that particular resource manager. For example, the scope of resource manager names may be limited to a single instance of the resource manager implementation, a single-system image, a single user, or a single collection of instances of the resource manager implementation.

Open Blueprint concatenated naming is an approach to achieving greater uniformity of these diverse namespaces by "plugging" existing resource manager-unique namespaces into the universal namespace (using a technique called "junctions"). With concatenated naming, a resource name is formed by appending a resource manager-specific name to a universal name. Figure 61 on page 76 shows two examples of concatenated naming.

Concatenated naming, though not as robust a solution as universal naming, can be accomplished with existing resources at lower cost and less disruption to current components, while still providing some distinct benefits:

- It provides a common "root" for all resource identification. Without this concept, specific knowledge of how to find the root of each resource-manager namespace is needed to locate all resources. With a common root, it is possible to build universal resource browsers that can "discover" a resource, even though the browser had no prior knowledge of its type or existence.
- It establishes a single concept of administrative domain scope and ensures that all resources have a consistent approach to referring to instances in "foreign" administrative domains.

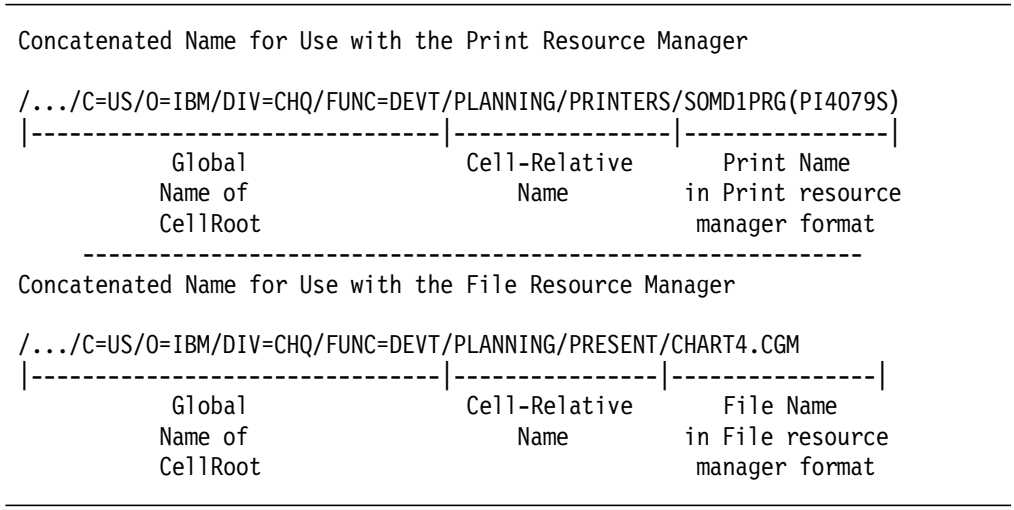


Figure 61. Concatenated Naming Example

3.2.3 Contextual Names

It is not always practical to use fully-qualified universal (or concatenated) names in every command, interface, or function. From a user perspective, such references would be time-consuming and error prone. From a programming perspective, such references require major changes in existing programming interfaces, and user interfaces. (A universal name can be up to 2048 characters long.)

Hence, it is common practice to use and support contextual names. A contextual name is a name that represents some portion of a fully-qualified name. Contextual names are valid only in a context in which the remainder of the fully-qualified name has been provided.

How context is established is currently an implementation choice of each resource manager. For existing resource managers, contextual names are often used to “map” their current programming interfaces to the universal name space. For example, some portion of a resource manager-specific name (such as a device name, disk name, or group name) can be treated as a symbolic reference to an environment variable that contains the remainder of the fully-qualified name.

Contextual names also provide a reference scope. For example, a contextual name may refer to the name of a cell, defaulting to the current cell if not otherwise specified. Setting this value may implicitly allow all subsequent operations to be performed with the cell of choice.

3.3 ISO Naming

ISO naming concepts include at least two aspects: (1) construction of names, and (2) registration of names. These concepts are described in several standards. This section provides an overview of ISO naming and registration concepts based on the ISO standards.

3.3.1 ISO 7498

In the field of information technology ISO concepts of naming are largely base on the basic reference model for open systems interconnection defined in ISO 7498. See Appendix A, “OSI Reference Model” on page 123 for a description of the OSI basic reference model. ISO 7498 Part 3, *Naming and addressing*, approaches the subject from the perspective of the OSI basic reference model.

This part of ISO 7498:

1. Defines general mechanisms for the use of names and addresses to identify and locate objects in the OSIE (Open Systems Interconnection Environment)
2. Defines the use of these mechanisms within the layered structures of the Basic Reference Model.

ISO 7498-3 defines a set of terminology for naming, including the following terms:

Primitive name A name that identifies an object and which is assigned by a designated naming authority. The internal structure of the name is not required to be understood or have significance to users of the name.

Descriptive name A name that identifies a set of one or more objects by means of a set of assertions concerning the properties of the objects of the set.

This distinction can be used to differentiate two conceptual approaches to identification of data objects— (1) an approach using simple identifiers that are not self-descriptive, and (2) an approach using structured, self-descriptive identifiers. For example, “306092288437” might be a hypothetical primitive name indicating a specific message routed over a network. “US.IBM.Office.Doc.RFTDCA” might be a hypothetical descriptive name indicating a specific category of document objects.

ISO 7489-3 explains that primitive names are unambiguously assigned to specific objects. Although primitive names are unambiguous, there may be more than one primitive name for the same object— that is, synonyms. Descriptive names, by contrast, consist of sets of assertions expressed in a formally defined language. A descriptive name may be incomplete, in that many objects satisfy its assertions; or complete, in that it identifies a single object. A primitive name can be a component of a descriptive name. Most of 7498-3 is devoted to a description of how names are used in associations between open systems based on the OSI basic reference model. The standard does not specify the syntax of specific names.

The following is an example of the role of naming (addressing) in the interaction between OSI systems:

“It is important to distinguish between the semantics of an (N)-address and the syntax used to represent an (N)-address within a given open system. (N)-addresses are passed across layer boundaries within open systems as parameters of (N)-service primitives. For (N)-service request/response primitives, the semantics of (N)-addresses are conveyed to the peer (N)-subsystem and passed across the layer boundary as parameters of the (N)-service indication/confirm primitives. Only the semantics of an (N)-address are conveyed by the (N)-service. The syntax of an (N)-address is a local issue and different representations may be used in different open systems.”

ISO 7498-3 refers to the use of directory functions provided by a directory facility— for example, to support name mappings. However, it is outside the scope of ISO 7489-3 to define a directory facility.

ISO 7489-3 establishes the concept of *naming-domains* and *naming-authorities*:

"Each naming-domain is administered by a naming-authority. A naming-authority is a registration authority, which only registers names and only has an administrative role. Although naming-authorities register the use of names, they do not participate in the binding of the name to an object. A naming-authority may register names itself or it may partition the naming-domain into naming-subdomains and delegate to a subdomain naming-authority the responsibility for naming within each naming-subdomain. The procedures for a naming-authority ensure the registration of unambiguous names and, if necessary, provide any rules that subdomain naming-authorities must comply with to meet the registration requirements."

In this connection, ISO 7489-3 establishes the idea that a naming hierarchy consists of a variable combination of *leaf* elements and *node* elements. A leaf, or terminal, element represents an actual name assignment. A node, or non-terminal element, represents a subsidiary naming authority that in turn can assign names within its own naming subdomain.

ISO 7489-3 recognizes that the operation of OSI naming requires the establishment of registration procedures, but it is outside the scope of ISO 7489-3 to define these procedures.

3.3.2 ISO 8824

ISO 8824 defines an abstract data description language useful for specifying data structures in a system-independent manner. For more information about ISO 8824, see Appendix C, "Overview of ASN.1" on page 135. The language is built on the concepts of data values and datatypes:

"The purpose of specifying a value is to distinguish it from other possible values. The collection of the value together with the values from which it is distinguished is called a *type*, and one specific instance is a *value* of that type. More generally, a value or type can often be considered as composed of simpler values or types, together with the relationships between them. The term *datatype* is often used as a synonym for type."

"The International Standard defines a notation which both enables complicated types to be defined and also enables values of these types to be specified. This is done without determining the way an instance of this type is to be represented (by a sequence of octets) during transfer. A notation which provides this facility is called a *notation for abstract syntax definition*."

"The purpose of this International Standard is to specify a notation for abstract syntax definition called *Abstract Syntax Notation One*, or ASN.1."

"This International Standard is supported by other standards which specify *encoding rules*. The application of encoding rules to the value of a type defined by ASN.1 results in a complete specification of the representation of values of that type during transfer (a transfer syntax)."

ISO 8824 is foundational to ISO naming concepts because it specifies a datatype called the *object identifier*. The ASN.1 object identifier is defined in ISO 8824 as one of the "built-in" datatypes along with others, such as the bitstring datatype and the integer datatype. In ISO 8824, the definition of object identifier is simply:

"A value (distinguishable from all other such values) which is associated with an information object."

The object identifier datatype has special semantics which are explained in ISO 8824:

"The semantics of an object identifier value are defined by reference to an object identifier tree. An object identifier tree is a tree whose root corresponds to this International Standard and whose vertices correspond to administrative authorities responsible for allocating arcs from that vertex. Each arc of the tree is labeled by an object identifier component which is a numeric value. Each information object to be identified is allocated precisely one vertex (normally a leaf) and no other information object (of the same type or a different type) is allocated to that same vertex. Thus, an information object is uniquely and unambiguously identified by the sequence of numeric values (object identifier components) labeling the arcs in a path from the root to the vertex allocated to the information object. Object identifier values contain at least two object identifier components."

"In general, an information object is a class of information (for example, a file format), rather than an instance of such a class (for example, an individual file). It is, thus, the class of information, (defined by some referenceable specification), rather than the piece of information itself, that is assigned a place in the tree."

ASN.1 object identifiers are often represented as a series of integers within braces as in the following example:

Example of ASN.1 OID

```
{ 1 0 8571 1 }
```

In this example, the leftmost component ("1") represents the numeric value assigned at the root node. The next component ("0") represents the numeric value assigned at the next subordinate node, and so on. The number of components, that is, the number of integers within braces, depends on how deeply the object identifier (OID) resides within the OID tree.

The concept of the OID tree was created by standards organizations, and OIDs are intended to support identification of information objects defined within international standards. The root node of the OID tree ID defined within the ASN.1 standard itself, ISO 8824, and the range of possible values at the root node all refer to standards organizations:

- Value of 0 refers to CCITT.
- Value of 1 refers to ISO.
- Value of 2 refers to joint ISO-CCITT.

Although at the top level of the OID tree the values refer to standards organizations, there is no reason why at lower levels values cannot be defined and assigned by other organizations wishing to identify their information

objects— for example, a software vendor who wishes to identify objects encoded in a proprietary file format.

3.3.3 ISO 9834

ISO 9834 Part 1 describes general procedures for the operation of OSI registration authorities. The CCITT analogue for ISO 9834 is the X.660 Recommendation. ISO 9834 establishes terminology for registration, including the following terms:

Registration The assignment of an unambiguous name to an instance of a type of information object in a way which makes the assignment available to interested parties

Registration agent An organization or a standard performing registration for instances of one or more types of information object

Registration authority An organization which acts as a registration agent.

The existence of an international standard devoted to registration procedures underscores the importance of registration in the establishment of a global naming system. Successful implementation of a global naming system requires both technically adequate naming structures and practical, viable administrative procedures for registering names.

Note from the preceding definitions that a registration agent does not have to be an organization. A registration agent can be an international standard itself. When a standard acts as registration agent, assignment of names is documented within the standard document and separate publication of a registry is not required. Obviously, assignment of global names within standards documents should be limited to naming domains with a low rate of change:

“Registration can be effected by a standard, by publishing in the standard the names and the corresponding definitions of instances of types of information object such a mechanism requires amendment of the standard for each registration, and hence is not appropriate in cases where the registration activity is high.”

“Alternatively, registration can be effected by permitting one or more organizations to act as registration authorities to perform registration on a flexible basis.”

ISO 9834 also establishes the term *structured name*:

Registration-structured-name A name which is unambiguous with the OSIE and which is assigned by registration.

The term *Registration-structured-name* is sometimes shortened to *R-structured-name* or simply *structured-name*.

ISO 9834-1 also recognizes the risk of synonyms:

“Synonyms are created when an instance of a type of information object is registered more than once. There may be valid reasons for creating synonyms, e.g., Directory aliases. It is difficult to detect occurrences of synonyms. In cases where synonyms are undesirable, it may be possible to reduce the number by such means as technical review or administrative fees (in the case of registration authorities). It must be decided in each case whether this is necessary and practical. There is no practical way to ensure that the same

instance of a type of information object has not been registered by multiple registration agents."

ISO 9834-1 establishes a hierarchy, or tree, of structured names. The root of this tree is the ISO 9834-1 standard itself. Each branch in this tree is identified by an integer value, and optionally by a *relative distinguished name (RDN)*, which is defined in ISO 9594. In ISO 9834 the RDN is described as follows:

"An RDN is a set of attribute value assertions. Each attribute value assertion is a sequence of two elements, an attribute type and an attribute value."

The structured name tree established by ISO 9834-1 is a subtree of the object identifier tree created by ISO 8824. It should be viewed as an implementation of the ASN.1 OID tree, and not as an alternative means of assigning identifiers to objects.

Since the structured name tree of ISO 9834 is a subtree of the ASN.1 OID tree, it must "plug into" the OID tree at some point. It plugs in by defining attributes that correspond to OID components in the highest level nodes in the ASN.1 OID tree.

The following example shows how a directory name can be constructed according to ISO 9834 that is equivalent to an object identifier constructed according to ISO 8824:

{ 1 0 8571 2 1 }

In this object identifier, the integer components have the following semantics:

1 "ISO" arc

0 "Standard" arc

8571 Identifies FTAM standard

2 Identifies a class of abstract syntaxes defined within the FTAM standard.

1 Identifies an abstract syntax for an information object named "pci."

To create a directory name for the pci information object, attributes named "ftam-components" and "abstract-syntax" would be created, and assigned values as follows:

{ standard = 8571, ftam-components = 2, abstract-syntax = 1 }

Thus, ISO 9834 establishes a complementary relationship between the RDN hierarchy and the OID hierarchy defined in ISO 8824:

"This Part of this International Standard (ISO 9834-1) assigns integer values to the top level arcs of the R-structured-name-tree in full accordance with ... ISO 8824. Responsibility for the subtrees beneath these arcs is also assigned in accordance with ISO 8824. An RDN is assigned in ... this Part of this International Standard (ISO 9834-1) for each of the paths from the root which are assigned values in ... ISO 8824. These RDNs provide unambiguous identification for these paths within the scope of the root."

An OID value defined according to ISO 8824 typically consists of a sequence of integers. The position of each integer indicates its node within the OID tree. An

RDN value defined according to ISO 9834 consists of a set of attribute value assertions, in which attributes can represent nodes in the OID tree, thus providing a correspondence between the two naming techniques.

ISO 9834 has defined the following directory name attributes corresponding to high-level nodes in the object identifier tree:

Table 1. Correspondence between Nodes and Attributes

OID Node	**Directory Attribute**
iso standard	standard
iso member-body	countryName
iso identified-organization	identified-organization
ccitt administration	administration
ccitt network-operator	network-operator
joint-iso-ccitt	joint-iso-ccitt

3.3.4 ISO 8879 and ISO 9070

ISO 8879, *Standard Generalized Markup Language (SGML),* contributes to development of global naming because it establishes the concept of *public text*.

"SGML specifies how markup, such as generic identifiers, attributes, and entity references, is recognized, but no specific GIs or other names are part of the language. The vocabulary is made up by users to meet their needs and is defined in the document type and link process definitions."

"Substantial benefit can be derived from individual use of SGML (as was the case with earlier generic coding and generalized markup language designs). However, trade organizations, technical societies, and similar groups desiring to interchange documents could benefit further by sharing document type definitions and other markup constructs."

"To this end, SGML includes a syntax for referencing text with public identifiers."

Public text in SGML is based on the entity feature, which permits portions of documents to be stored externally to the main part of the document. Externally stored text portions must be referenced in a consistent manner, and SGML defines an external identifier for this purpose. An external identifier is introduced by the keyword SYSTEM or the keyword PUBLIC. The SYSTEM keyword signifies that the data portion is named within the local processing system, and the external identifier contains appropriate identification information that is meaningful to the local system. The keyword "PUBLIC" introduces an external identifier that contains a *Formal Public Identifier,* which in turn contains an owner identifier and a text identifier. A Formal Public Identifier is intended to identify a text portion in public contexts, and can be used in industry, national, or international namespaces.

The relationship of ISO 9070 to SGML is indicated in its title, *Information processing-SGML support facilities-Registration procedures for public text owner identifiers.* ISO 9070 further develops the concept of public text. It further specifies the syntax of public identifiers and describes registration procedures for authorities that register public text.

3.3.5 Ancillary ISO Standards

The standards discussed in this section are independent international standards in their own right. The are not parts of standards previously discussed. Nevertheless, they provide important support for naming systems described in other standards.

3.3.5.1 ISO 6523

ISO 6523, *Data interchange-Structure for the identification of organizations,* supports global naming by providing a means to identify organizations. ISO 6523 defines a Structure for the Identification of Organizations (SIO).

The Structure for the Identification of Organizations consists of three components:

1. International Code Designator (ICD)
2. Organization code
3. Organization name.

The International Code Designator is a four-digit code that identifies the authority issuing the organization code. The organization code consists of 1-14 characters that uniquely identify an organization within an organization coding scheme. The organization name is a text string up to 250 characters in length.

For more information about the ISO 6523 method of identifying organizations, see Appendix D, “Overview of ISO 6523” on page 143.

3.3.5.2 ISO 639

ISO 639, *Codes for the representation of names of languages,* specifies codes for languages—for example, “en” for English.

3.3.5.3 ISO 3166

ISO 3166, *Codes for the representation of names of countries,* specifies codes for countries—for example, “US” for United States of America.

3.4 ASN.1 Object Identifiers

One approach to global naming involves the use of ASN.1 object identifiers. Abstract Syntax Notation One (ASN.1) is an internationally standardized data representation language. It is defined in ISO 8824, *Information Processing- Open Systems Interconnection—Specification of Abstract Syntax Notation One (ASN.1).* For more information about ASN.1, see Appendix C, “Overview of ASN.1” on page 135.

ASN.1 was created to support the OSI reference model. See Appendix A, “OSI Reference Model” on page 123 for a description of the OSI basic reference model.

3.4.1 Role of ASN.1 Object Identifier

The ASN.1 OID has a tag of universal class, number 6. The value of an ASN.1 OID is a list of components. There are four kinds of possible components:

1. NameForm
2. NumberForm
3. NameAndNumberForm
4. PrefixedNumberForm.

NameForm components are limited to values specified in the ASN.1 standard itself. NumberForm components may be integer values or references to other ASN.1 OIDs. The NameAndNumberForm type of component is used when the creator of a numeric value for an ASN.1 OID wishes to associate another identifier with the numeric value. The PrefixedNumberForm is used when a NameForm value acts as a prefix followed by NumberForm values.

What is the meaning of the numbers that typically occur in an ASN.1 OID? The numbers have values according to the ASN.1 *object identifier tree.* The following description, taken from ISO 8824, explains the OID tree:

"The semantics of an object identifier value is defined by reference to *an object identifier tree.* An object identifier tree is a tree whose root corresponds to this International Standard and whose vertices correspond to administrative authorities responsible for allocating arcs from that vertex. Each arc of the tree is labelled by an object identifier component which is a numeric value. Each information object to be identified is allocated precisely one vertex (normally a leaf), and no other information object (of the same or a different type) is allocated to that same vertex. Thus an information object is uniquely and unambiguously identified by the sequence of numeric values (object identifier components) labelling the arcs in a path from the root to the vertex allocated to the information object."

"An object identifier value is semantically an ordered list of object identifier component values. Starting with the root of the object identifier tree, each object identifier component value identifies an arc in the object identifier tree. The significant part of the object identifier component is the 'NameForm' or 'NumberForm' which it reduces to, and which provides the numeric value for the object identifier component."

Branches in the ASN.1 object identifier tree are typically referred to as arcs.

3.4.2 Examples

The following example of an ASN.1 OID identifies an information object defined in ISO 8571 (FTAM).

{ 1 0 8571 2 1 }

This is an example of the NumberForm type of OID. This OID can also be represented as a NameAndNumberForm type of OID. The ASN.1 standard assigns text values to certain numeric values; for example, "1" in the leftmost position can be replaced with "ISO." Substituting text where allowed, the previous OID can be represented as follows:

{ iso standard 8571 abstract-syntax(2) ftam-pci(1) }

Information objects referred to by the expression “ftam-pci(1)” are defined in the FTAM standard.

The following example illustrates the PrefixedNumberForm of OID:

First, define “ftamProtocol.”

ftamProtocol OBJECT IDENTIFIER ::= {iso standard 8571 }

Then use “ftamProtocol” as a prefix.

{ ftamProtocol 2 1 }

ASN.1 OIDs typically consist of numeric values. However, a limited number of NameForm values are specified within ISO standards and CCITT Recommendations. The following section describes NameForm values specified in ISO 8824. In the following table captions, “C” refers to the leftmost component within an OID. Thus, an OID can be described as:

{ component C component C+1 component C+2 (etc.) }

Table 2. Values for Component C

NumberForm Value	NameForm Value	Meaning of Value
0	CCITT	Arc of OID tree originates with CCITT
1	ISO	Arc or OID tree originates with ISO
2	Joint-ISO-CCITT	Arc of OID tree originates with both ISO and CCITT

Table 3. CCITT Arc: Values for Component C+1

NumberForm Value	NameForm Value	Meaning of Value
0	Recommendation	Arc defined by CCITT Recommendation
1	Question	Arc defined by CCITT Question
2	Administration	Arc defined by CCITT Administration
3	Network-operator	Arc defined by CCITT Network-operator

Table 4 (Page 1 of 2). ISO Arc: Values for Component C+1

NumberForm Value	NameForm Value	Meaning of Value
0	Standard	Arc defined by ISO Standard
1	Registration-authority	Arc defined by ISO registration authority
2	Member-body	Arc defined by ISO member body

Table 4 (Page 2 of 2). ISO Arc: Values for Component C+1		
NumberForm Value	**NameForm Value**	**Meaning of Value**
3	Identified-organization	Arc defined by ISO identified organization

3.4.3 Subsequent Arcs – CCITT

Component C+2 in the “Recommendation” arc has a NumberForm value 1 through 26 or a NameForm value a through z, identifying the series to which the Recommendation belongs; for example, X.400 is in the X series. Component C+3 has the number of the Recommendation. Values for subsequent components are assigned within the Recommendations.

Component C+2 in the “Question” arc represents a CCITT Study Group and Study Period according to the following formula:

study group number + (period * 32)
where “period” has the value 0 for 1985-1988, 1 for 1989- 1992, and so on.

Component C+3 represents individual Questions, and components beyond this are assigned as necessary by working groups assigned to each Question.

Component C+2 in the “Administration” arc has values assigned per Recommendation X.121. These values represent national telecommunications administrations. Subsequent components are assigned as necessary by administrations.

Component C+2 in the “Network-operator” arc also has values assigned per Recommendation X.121 to represent networks, and subsequent components are assigned as necessary by administrations.

The U.S. State Department is the official administrator of the CCITT arc in the U.S. However, the State Department has formally delegated this responsibility to ANSI.

3.4.4 Subsequent Arcs – ISO

Component C+2 in the “Standard” arc has a NumberForm value representing the number of an ISO Standard. For example, “8571” can represent the FTAM standard. If the standard is a multi-part standard, then Component C+3 is the part number. Components beyond the standard number and part number, if applicable, are assigned as necessary within the standard.

Component C+2 in the “registration authority” arc identifies an organization recognized by ISO as a registration authority for identification of information objects. The meaning of subsequent components is determined as necessary by the registration authority.

Component C+2 in the “member body” arc has a numeric value representing the ISO country code assigned to the member body; for example, the United States has ISO country code 840. Subsequent components are assigned as necessary by the member body.

Component C+2 in the “identified organization” arc has a numeric value corresponding to an *International Code Designator* number assigned in

accordance with ISO 6523. ISO may endorse an organization as an International Code Designator (ICD), giving that organization the right to issue ASN.1 OID values to other organizations in turn. For example, ISO has recognized IBM as ICD number 18.

3.4.5 Subsequent Arcs – Joint ISO and CCITT

The meaning of subsequent components in the "joint-ISO-CCITT" arc is defined as necessary when standards are developed jointly by ISO and CCITT. For example, ISO 9834 further describes the "joint-ISO-CCITT" path within the object identifier tree, and ISO 9594 defines subordinate directory-related object identifiers under the joint-ISO-CCITT arc.

3.4.6 Use of ASN.1 OIDs

ASN.1 object identifiers can be used in more than one way. OIDs can be used:

1. To identify an organization. Used this way, OID components represent a hierarchy of organizations that can help identify an information object from an administrative or organizational perspective.
2. To identify an object class. Used this way, OID components can represent superclasses, classes, and subclasses of objects, helping to identify an information object by showing to what classes it belongs.
3. To identify an object instance. In this approach each instance of an object within a class is assigned its own number, which becomes the rightmost component in the OID. Used this way, OID components can provide globally unique identifiers for specific information objects— so to speak, "global serial numbers."
4. In combinations of the above.

3.4.7 Example

The following example describes a proposal made by a major North American vendor of desktop software concerning possible use of ASN.1 OIDs to provide global identification of encoded word processing documents for heterogeneous interchange.

The proposal begins with OID components identifying organizations only:

{ 1 2 840 113556 }

Table 5. Analysis of Proposed OID

Leftmost Component(C)	**Component C + 1**	**Component C + 2**	**Component C + 3**
1	2	840	113556
Identifies ISO	Identifies ANSI	Identifies USA	Identifies software vendor

This portion of the OID identifies a path in the OID tree beginning with ISO, then selecting the ISO member-body path, then selecting USA, then selecting an organization, based in the USA, which has been assigned the value 113556. This organization has created subsequent OID components as follows:

- Application software— value 5
- Specific word processor product— value 2

- Current versions of this product— values 2, 3, 5.

If this proposal were implemented, encoded document files produced by this product might contain encoded ASN.1 OIDs representing the following components:

{ 1 2 840 113556 5 2 5 }

Thus, the OID would provide a globally unique “tag” labelling each document file. In a heterogeneous environment, other software products might interrogate this data field, providing they knew where to look for it in the data. If receiver products recognized the encoded OID, they could determine whether they were capable of processing the document.

Should ASN.1 OIDs be used to identify individual information objects? The answer depends on what type of identification is most appropriate for the user. For example, does a user require a simple identification number—such as a serial number for a piece of equipment? Or does a user require a self-describing identifier that might be considered a name of some kind? An example of OID usage for instance identification is provided by the Internet, which uses OIDs to identify instances of Simple Network Management Protocol (SNMP) objects.

The encoded form of an ASN.1 OID is a flat number. The abstract form, with is seen in the following example, can be parsed in a straightforward manner by a human interpreter:

{ 1 0 8571 2 1 }

If a person understands the meaning of the components within the braces, the significance of this OID is clear.

The abstract form, however, only appears in documentation intended for human readers. When ASN.1 OIDs are encoded for machine storage or transmission of data, they appear as flat numbers, and their semantic meaning is not apparent. For an example of an encoded ASN.1 OID, see 3.4.10, “Encoding an ASN.1 OID” on page 90. Thus, in actual use ASN.1 OIDs function more as simple “ID numbers” than as descriptive names. This may be optimal for machine processing— for example, for straight table lookups. But it may be disadvantageous for applications where data identifiers must be interpreted and processed directly by humans.

Another consideration is that components within an OID provide a form of untyped naming. That is, even for a “decomposed” OID, the semantic meaning of the components is not always apparent. At the highest level of the OID tree, the meaning is apparent—for example, in the leftmost position there are only three possible values—0, 1, or 2— with the meanings 0 = CCITT T; 1 = ISO; 2 = Joint-ISO-CCITT. But as components are added, the total semantic meaning gradually becomes less apparent, and establishing the meaning of all components in an OID may require referencing multiple standards documents and registry documents.

3.4.8 IBM Use of OIDs

How is IBM identified within the OID identification tree? The strategic identification for IBM for future use is represented by the following components:

{ 1 3 18 }

These components signify a path beginning with ISO, selecting the “identified organization” path, and then identifying IBM. The identified organization path is described in ISO 6523, and grants approved organizations the right to create subsequent coding structures or subtrees for other organizations. IBM has been recognized by ISO as an “identified organization” and has been assigned the number 18. An identified organization can be an International Code Designator (ICD), and the value 18 is IBM′s ICD number.

U.S. organizations do not have to register as ICDs. Another possible path is:

{ 1 2 840 }

This path begins with ISO, selects the member-body path, and then selects the U.S. Any U.S. organization may request an identification number within this path, and ANSI is the registration authority that assigns identification numbers within this path. However, this ISO path may be deprecated in favor of the joint-ISO-CCITT path described in the following paragraph.

Recently, another path has been proposed by which U.S. organizations can register in the OID tree:

{ 2 16 840 1 }

In this path, “2” represents the joint-ISO-CCITT path, “16” represents the country path, “840” represents the U.S., and “1” represents ANSI. Following the “1,” would appear the unique registration number assigned by ANSI to any U.S. organization for registration in the OID tree.

3.4.9 Subtrees in IBM

IBM has established two OID subtrees within the { 1 3 18 } path.

Table 6. IBM OID Subtrees for 1 3 18 Path

IBM ICD Path	Subtrees	Purpose
1 3 18	1 3 18 0	IBM objects registry
1 3 18	1 3 18 16383+	Network identifiers for customers

IBM Objects Registry: IBM has created six subtrees within this path.

Table 7 (Page 1 of 2). IBM Internal Subtrees

IBM Path	Purpose/Owner
1 3 18 0 0	Network Management Raleigh
1 3 18 0 1	Fonts/Boulder
1 3 18 0 2	Directory/Toronto

Table 7 (Page 2 of 2). IBM Internal Subtrees	
IBM Path	**Purpose/Owner**
1 3 18 0 3	DDM/Rochester
1 3 1 8 0 4	Distributed Print/Boulder
1 3 18 0 5	Office Objects/Westlake

Network identifiers for customers: In its role as ICD, IBM has the capability to administer numeric identifiers for customers. Such identifiers have integer values greater than 16383, and are assigned when required to register customer network identifiers. These identifiers are limited to those already registered in the SNA Network ID Registry.

3.4.10 Encoding an ASN.1 OID

ASN.1 OIDs are assigned and documented in an abstract, human-readable form, typically in the following manner:

{ component1 component2 component3... componentN }

where components are typically decimal numbers. When abstract ASN.1 OIDs are converted to binary, they are encoded according to rules specified in ISO 8825, the Basic Encoding Rules for ASN.1 (BER). Theoretically, additional sets of encoding rules for ASN.1 can be standardized. However, in practice BER is the recognized method of encoding ASN.1. For a full understanding of the BER rules, please consult ISO 8825.

The following example illustrates how the IBM OID for an RFT document is encoded.

OID Encoding Example

The decimal representation of the OID appears as follows:

{ 1 3 18 0 5 0 50950 11 }

When converted according to BER, this identifier yields the following binary values:

Binary String	Meaning
00000110	ASN.1 Object Identifier follows.
00001001	The length of this OID is 9 bytes (in ISO 8825 bytes are referred to as *octets*).
00101011	Beginning of OID tree— indicating ISO path and Identified Organization path.
00010010	Identified Organization is IBM (the number assigned to IBM is 18).

Example continued on next page

OID Encoding Example

Continued from previous page.

Binary String	Meaning
00000000	Network Systems Object Class— 0.
00000101	Office Object Class— 5
00000000	DIA Code Point— 0
10000011	The first bit of this octet is a flag indicating that this is not the last octet representing the seventh sub-identifier. Remaining bits are combined with subsequent octets representing seventh sub-identifier.
10001110	The first bit of this octet is a flag indicating that this is last octet representing the seventh sub-identifier. Remaining bits are combined with preceding octets representing the seventh sub-identifier. Decimal value of seventh sub-identifier is 50950.
00001011	The first bit of this octet is a flag indicating that this is last octet (in this case, only octet) representing the eighth sub-identifier. The decimal value of the eighth sub-identifier is 11.

The hexadecimal representation of the complete encoded OID (with spaces added for clarity) appears as follows:

06 09 2B 12 00 05 00 83 8E 06 0B.

Software for converting ASN.1 data into BER form is available in the industry and within IBM.

3.5 OSF Naming, CORBA Naming, and Naming APIs

Other groups have also developed naming schemes and in the following sections will the OSF and CORBA concepts be discussed as well as the activities to establish a consistent naming structure for Application Programming Interfaces (API).

3.5.1 OSF Naming

The Open Software Foundation (OSF) is developing a software environment for open computing called the Distributed Computing Environment (DCE). DCE currently consists of the following components:

- DCD Threads
- DCD Remote Procedure Call (RPC)
- DCE Directory Service
- DCE Distributed Time Service (DTS)
- DCE Security Service
- DCE Distributed File Service (DFS)
- DCE Diskless Support Service.

DCE also includes a system management component that coordinates administrative functions for other components.

Because DCE supports distributed computing, the issue of naming within variable scopes is important. The issue of distributed naming is addressed primarily in the DCE Directory Service component. The DCE Directory Service is essentially a distributed, replicated database service providing storage and access to arbitrary information attributes associated with system entities, such as users, machines, and data objects, that may be geographically dispersed. The DCE Directory Service has three functional components:

- Cell Directory Service (CDS)
- Global Directory Service
- Global Directory Agent (GDA).

The Cell Directory Service supports storage of resource names and attributes within a local DCE cell, and is optimized for local processing. The CDS represents a naming service within a restricted namespace— the local DCE cell.

The Global Directory Service supports the directory function above the cell level and operates in the global namespace. It is invoked when a local cell requests access to a directory object not contained in the CDS. GDS functionality is based on ISO 9594, *The Directory,* and its CCITT analogue, the X.500 series of Recommendations. DCE also supports the Domain Name Service (DNS) of the Internet for access to entities outside the local cell.

The Global Directory Agent provides the interface between the CDS and either the GDS or the DNS. It services local cell directory requests that cannot be satisfied locally and determines whether GDS or DNS must be invoked. Programmatic access to GDA services is available through the XDS set of directory APIs specified by X/Open and X.400 Application Programming Interface Association (XAPIA).

The DCE namespace is defined to be the set of all names used by the DCE Directory Service. It is hierarchical and supports naming structures employed by GDS and DNS. Global names begin with the characters “/...” and are registered in GDS or DNS.

CDE provides the capability to generate and register Universal Unique Identifiers (UUIDs). These can be used in a transient manner— for example, to identify RPC interfaces in a session. They can also provide persistent names for cell objects. If necessary, recognition of UUIDs outside the DCE context might be feasible by linking appropriate UUIDs to ASN.1 OIDs.

3.5.2 CORBA Naming

The Object Management Group (OMG) has developed the Common Object Request Broker Architecture (CORBA) specification to support an object-oriented approach toward the realization of open systems. The naming philosophy in CORBA is that an object may have multiple names meaningful in multiple namespaces with appropriate name mapping provided by the system. Within each namespace, an object name should be unambiguous. Such names are considered object attributes and are subject to change. In addition to such context-sensitive, changeable names, an object handle can be assigned to each object that is unique and not dependent on other names.

The OMG Object Model also establishes the concept of *object identifier*, or OID. Despite the acronym, this is not the same as the ASN.1 object identifier (OID) defined in ISO 8824. The CORBA OID is a unique identifier for an object that is distinct from and independent of the object′s characteristics. Object characteristics can change but object identity cannot. The OMG Object Model establishes the concept of OID but does not specify the syntax of this construct nor how it is to be implemented. The internal structure of the OID is "opaque" to the Object Model.

Object Request Broker (ORB) Name Services provide name mapping from the requester name domain into the object method domain. Such names are not required to be unique or universal.

3.5.3 Naming APIs

The growing importance of consistent naming across the global namespace has generated a request for definition of consistent naming of Application Programming Interfaces (APIs) within the X/Open community. Definition and acceptance of consistent APIs can facilitate movement toward open systems in several aspects of information technology, and X/Open has published many specifications with this objective. Examples include XDS (APIs for directory services), XOM (APIs for handling ASN.1 abstract data types), and XTI (APIs for transport services).

X/Open is considering launching a technical effort to define a new set of APIs to provide consistent programmatic access to a variety of naming systems. The name for this proposed project is *Federated Naming Interface*.

The proposal is based on the following rationale:

"There is a great degree of diversity and incompatibility amongst existing naming systems, due in part to different requirements, such as scalability, performance, granularity, autonomy in naming policy, and in part because naming is often tightly embedded in specific services, such as the file system, and presented only through that service′s interfaces. Not only do the naming interfaces differ widely, but the essential naming operations are obscured."

"At present, there is no naming interface that includes the basic naming operations that any naming service can support. Therefore, applications use the interface of a specific service that embeds the naming functionality. This means that the application is not portable to an environment where the equivalent naming functions are provided by a different service. Even within the same environment, it is not easy to replace the underlying naming service with a different one. The resulting impediments to building well-integrated distributed systems are substantial."

"Most applications that were originally conceived and developed for single machine environments use only a single naming system. Even when these applications are evolved to work in a distributed environment, they have very limited access to objects in the network. Historically, a small percentage of applications use composite names— names that span multiple naming systems— for accessing remote objects."

"A federated naming system is an aggregation of autonomous naming systems that cooperate through a standard interface to implement name resolution for composite names. The member naming systems can directly resolve composite

names without using intermediate clerks, agents or gateways. The standard naming interface to clients allows the use of composite names."

Chapter 4. Time Services Technologies

Time keeping and timer services are generally highly integrated with the products they support. They are generally local in nature and are incompatible between different types of systems. There are ad hoc ways of converting time formats between systems.

A distributed computing system has many advantages but also brings with it new problems. One of them is keeping the clocks on different nodes synchronized. In a single system, there is one clock that provides the time of day to all applications. Computer hardware clocks are not completely accurate, but there is always one consistent idea of what time it is for all processes running on the system.

In a distributed system, however, each node has its own clock. Even if it were possible to set all of the clocks in the distributed system to one consistent time at some point, those clocks would drift away from that time at different rates. As a result, the different nodes of a distributed system have different ideas of what time it is. This is a problem, for example, for distributed applications that care about the ordering of events. It is difficult to determine whether event A on node X occurred before event B on node Y because different nodes have different notions of the current time.

This chapter discusses how time is handled for distributed systems:

- Time in the IBM Open Blueprint
- Time in IBM′s MVS
- Time in OSF DCE

4.1 Time in IBM′s Open Blueprint

The TIME resource manager maintains knowledge of the time of day and synchronizes the system clocks in the distributed system to a limited, but known, degree of accuracy. Whenever the accuracy of a local clock′s time is beyond acceptable tolerance, time clients solicit the time from several time servers within the network. The local clock is then adjusted based on the intersection of the answers received from the time servers, allowing for processing and network transmission delays. Time servers synchronize with each other so that arbitrarily large networks can be synchronized.

Some time servers have access to authoritative QUALITY TIME PROVIDERS (such as a radio signal from a national standards organization), so that even networks that are not interconnected are mutually synchronized. All quality time providers in the world hold consistent time values. Therefore, the time resource manager would never attempt to adjust the clock of a quality time provider.

Adjusting a clock always takes the form of slightly changing its apparent tick rate, so that the clock gradually comes into synchronization, avoiding local requesters observing a discontinuity, and, especially, avoiding the appearance of the clock running backwards.

4.2 MVS Time and Time Services

MVS is IBM′s proprietary operating system for large host systems and is not usually considered to be a distributed system. However, many installations have several MVS systems sharing data and work. These sharing systems can be located either quite close together or can be far apart. The manner in which they are connected can also vary immensely. They can be connected by remote lines or via channel-to-channel connectors or just by sharing DASD storage.

For most of these connections, there are no formal ways to exchange time or to guarantee that times on one system are within a certain tolerance of times on another system in the complex. Traditionally, installations have had to use manual methods to keep different systems that were coupled in this manner in sync. This has meant that the applications were not able to demand a high degree of guaranteed time synchronization between systems.

MVS/ESA Version 4 introduced the concept of a *sysplex* to MVS. A sysplex consists of a number of MVS systems that are coupled either by channel-to-channel connections or a coupling facility (from MVS/ESA Version 5). For many purposes, the systems comprising a sysplex may be treated as a single system. One of the characteristics of a sysplex is a common time source for all of the systems in the sysplex. It is now possible to be very confident that times recorded on a member of a sysplex are the same as times recorded on any other member. This opens up the possibility for applications to maintain logs and other time-dependent files on several systems and be able to merge them and know that the order in which events were recorded in all of them was correct. For a detailed discussion of MVS time, see Appendix B, “Time in MVS” on page 127.

4.3 DCE Distributed Time Services (DTS)

Many applications use timestamps to coordinate independent events. The Distributed Time Services (DTS) runs on every host in the Distributed Computing Environment, keeping host clocks closely synchronized. It is also possible to synchronize the distributed environment with others by connecting to an external time signal (like a *Universal Time Coordinated* provider). The position of DTS in the DCE hierarchy of services can be seen from the shaded box on Figure 62 on page 97.

The components of DTS are :

- Time Clerk
- Time Servers
  - Local Time Server
  - Global Time Server
  - Courier Time Server
  - Backup Courier Time Server
- Time Representation
- DTS Application Programming Interface
- Time Provider Interface (TPI).

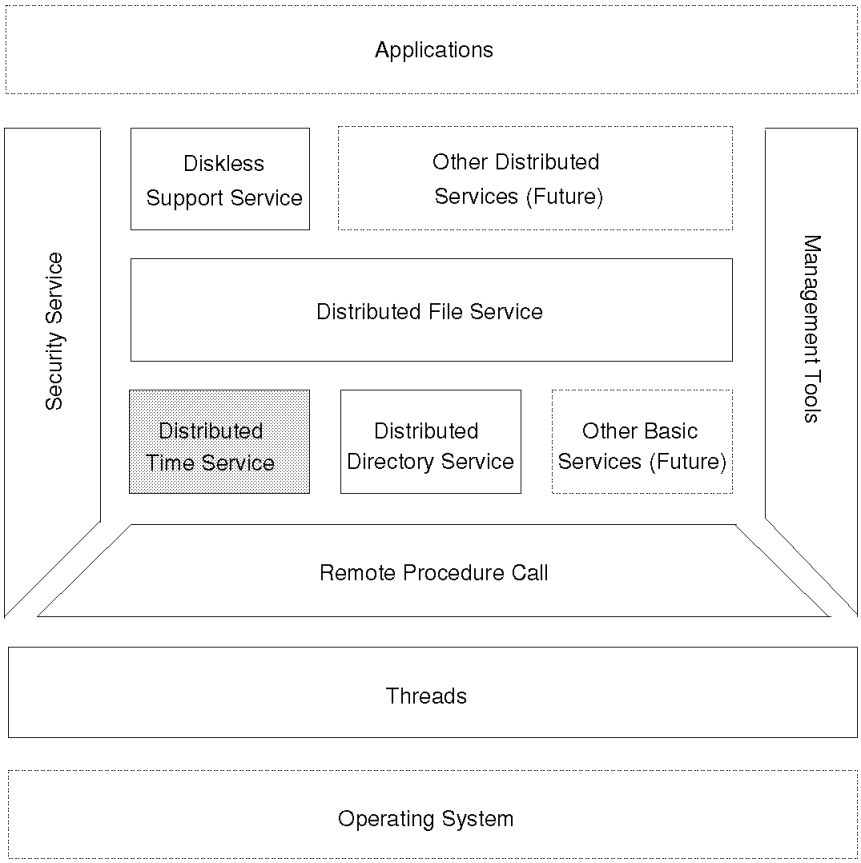


Figure 62. Distributed Time Services as a DCE Component

4.3.1 Why a Time Service

Problems can occur when clocks are not synchronized. On networks composed of multiple hosts, each host has its own clock and it own time reference.

A good example of troubles due to unsynchronized clocks is the UNIX `make` command. The `make` command selects files for compilation based on their creation time. Modified source files on a tardy host can appear older than corresponding object files on the host with the faster clock. In this case, `make` does not recompile files that should be recompiled.

DTS helps to avoid such situations by synchronizing host clocks in LANs and WANs with the following:

- DTS provides a way to periodically synchronize the clocks on the different hosts in a distributed system.
- DTS also provides a way of keeping that synchronized notion of time close to the *correct* time. In DTS, a *correct* time is considered to be *Universal Time Coordinated* (UTC), an international standard.

4.3.2 How Does It Work ?

If we assume that every server on the network has an accurate clock, any client can ask any server for the correct time. A delay is involved both in propagating the request to the server and in getting the response from the server.

A perfectly accurate time source does not exist. All sources are only approximately correct.

4.3.2.1 Client/Server Synchronization

Figure 63 depicts the time sequence that is used when synchronizing a client clock with the time server clock.

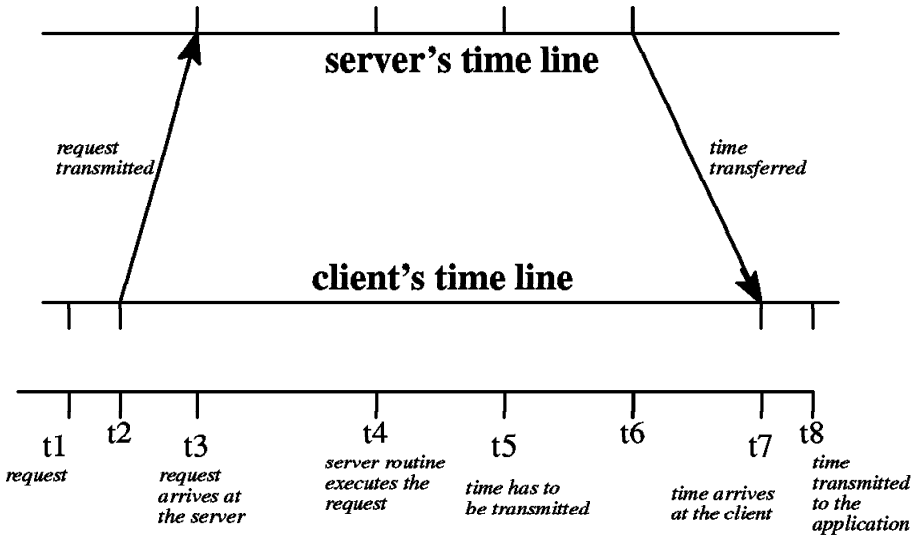


Figure 63. Client/Server Time Synchronization

The client wants to update its clock by obtaining the correct time from the server:

- At time *t1*, the client application makes the request to query the server for the time of day.
- At time *t2*, this request is transmitted.
- At time *t3*, the request arrives at the server.
- At time *t4*, the request is passed to the server routine that deals with time, and the clock is read.
- At time *t5*, this routine requests that the time be transferred to the client.
- At time *t6*, the time is transferred.
- At time *t7*, the time arrives at the client.
- At time *t8*, the time is finally received by the original application.

Let *S(t4)* be the value of the server's clock at time *t4*. The client has to get *S(t8)*. Let *C(tx)* be the value of the client's clock at time *tx*. The time that has elapsed between when the client made its request and when it received the response is *C(t8)-C(t1)*. Thus what the client learns from this transaction is that *S(t8)* is in the range [*S(t4),S(t4)+(C(t8)-C(t1))*].

There are two problems with this computation. Neither the client nor the server clock is continuous. Instead, they advance in discrete steps, called ticks. The length of each tick is called the clock *resolution*. Assume the length of a tick to

be *tl*. If *t1* occurs just after a tick and *t8* occurs just before a tick the actual value of *t8-t1* could be as great as *C(t8)-C(t1)+tl*. The second problem is that the client's clock may drift over time. If we assume that this drift rate is bounded by a constant Γ, the upper bound on *t8-t1* is $(C(t8)-C(t1)+tl)(1+\Gamma)$ so we can have the value of *S(t8)* estimated as $[S(t4), S(t4)+(C(t8)-C(t1)+tl)(1+\Gamma)]$.

Note: *tl* could be around ten milliseconds, the communication delay might be 20 milliseconds and Γ could be .0004 milliseconds per second.

4.3.2.2 Inaccuracy of a Clock

The *inaccuracy* of a clock is bound on the error of a clock as a function of time; the clock does not tell us precisely what time it is, but rather the time lies somewhere within an interval.

The *inaccuracy* of a clock changes with time. We can place a bound (the *drift*) on the rate of change and assume that it is constant.

We assume that a clock always progresses forward. If each tick represents the passage of *tl* seconds, *tl* is measured by the clock, but we must know how much it is in *real time*.

If *t1* and *t2* represent two instants in real time, *t2-t1* is not greater than $T(t2)+I(t2)-(T(t1)-I(t1))$. So drift is not greater than $T(t2)+\Gamma \cdot (t2-t1)+2*I(t1)-T(t1)$.

If we let $T(t2)-T(t1)$ be *tl*, the clock resolution, and we assume that *T(t1)* is the correct time, then *t2-t1* is no greater than $tl+\Gamma \cdot (t2-t1)$.

So the duration of a time interval, *t2-t1*, required to insure that the time measured by the clock is at least one clock tick (*tl*) is no greater than $tl/(1-\Gamma)$.

4.3.2.3 How the Client gets the Correct Time

To improve its chance to get the *real time* the client has to consult several servers and then somehow arrive at an average.

The servers reply with what it estimates to be the correct time and also with a bound on the accuracy of this time estimate.

Thus, the client receives a set of intervals with which it has to determine a current time interval that is tighter than any intervals obtained (Figure 64).

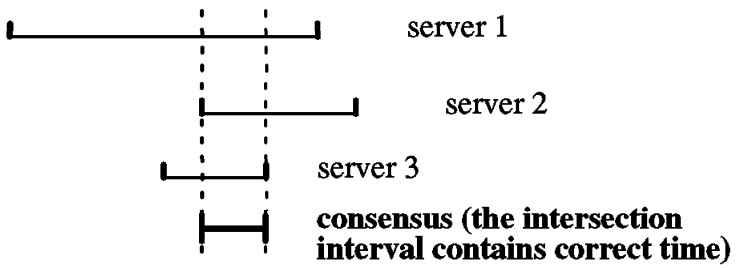


Figure 64. Time Intervals

One or more of the servers might be totally wrong about the time of day. In this case, the intersection of the interval received by the client might be null.

If we assume that there is at most one faulty server (Figure 65 on page 100), then any point contained in at least three of the intervals can represent the correct time.

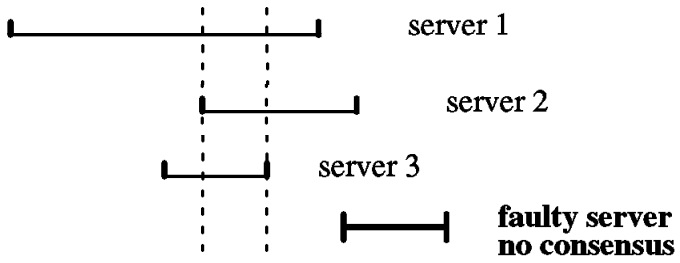


Figure 65. Faulty Server

4.3.2.4 Universal Time Coordinated (UTC)

The Universal Time Coordinated (UTC) is, by international agreement, based on atomic clocks. However, also by international agreement, UTC is occasionally adjusted so that UTC is not terribly different from the observed time at the Greenwich meridian. This adjustment is performed by addition (or subtraction) of *leap seconds*, which may be added to (or subtracted from) UTC at the end of any month.

Since it is not known sufficiently far in advance when these *leap seconds* will occur, the DCE Distributed Time Service does not have built it knowledge about when to apply *leap seconds*.

However, since a *leap second* might occur at the end of any month (and since the clock uses UTC), we must add one second to the inaccuracy of our clocks at the end of each month. The added inaccuracy induced by a possible *leap second* will be dealt with by some external, knowledgeable source.

4.3.2.5 Time Clerk

The time clerk is the client side of the DTS. It runs on a client machine and keeps the machine's local time synchronized by asking the Time Server for the correct time and keeps the local clock reasonably accurate.

In a correctly configured cell, the intersection of the time intervals of each clerk and server will not be null. However, it is desirable that each clerk (and server) have relatively small inaccuracies. It is easy, but not useful, to have all clocks in agreement only for a reason of large inaccuracy.

The clerk maintains the time to a desired inaccuracy (the default maximum inaccuracy is .1 seconds). First the clerk contacts a set of servers to obtain the time with an improved accuracy.

To prevent all clerks from synchronizing their clocks at the same time, if the current time *T0* and *T* is the amount of time that will elapse until maximum inaccuracy is reached, then the time at which resynchronization will take place is chosen randomly within the interval [*T0+T/2,T0+T*].

When a clerk synchronizes its clock, it attempts to contact at least three servers. It first tries three local servers. If it cannot make contact with enough local servers, the clerk contacts the global servers.

The clerk maps the server's time into the clerk's time frame, increasing the inaccuracy to take into account uncertainties in when the server obtained the time.

4.3.2.6 Time Servers

A time server is a node that is designated to answer queries about time. See Figure 66.

There are two different kind of servers: the *local time servers* and the *Global Time Servers*. A collection of computers that are close in terms of communication delay (a LAN) need a set of local time servers to keep the time. These servers periodically synchronized their clocks with one another's.

More frequently, clerks synchronize with at least a subset of the local time server's clocks, and possibly, with some global servers.

The *Global Time Servers* are time servers in other LANs (within the cell) that clerks and local time servers may contact if additional sources of time information are needed.

The local time servers are, with respect to the communication time, near one another and the clerks, while *Global Time Servers* are somewhat farther away.

The algorithm used to synchronize a server's clock with the clocks of other servers is essentially the same as the algorithm just discussed for clerks.

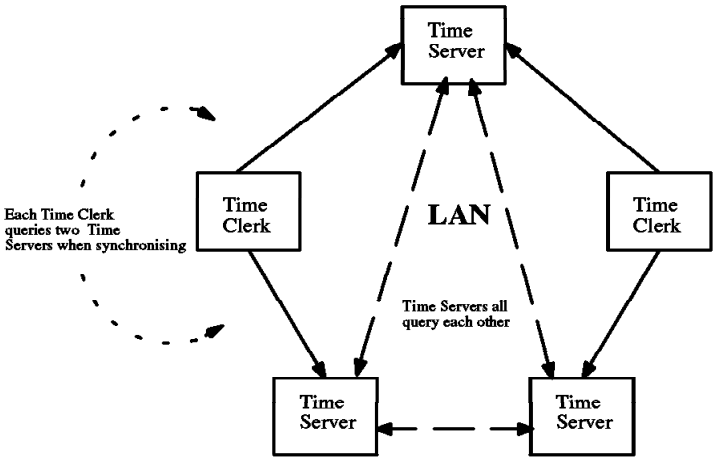


Figure 66. Time Servers

4.3.2.7 Courier Server

A server must be synchronized with at least *minServers-1* other servers, but it will try to synchronize with as many servers as there are on the LAN.

If there are fewer than *minServers* on the LAN, a server designated as *Courier* always attempts to contact at least one global server. See Figure 67.

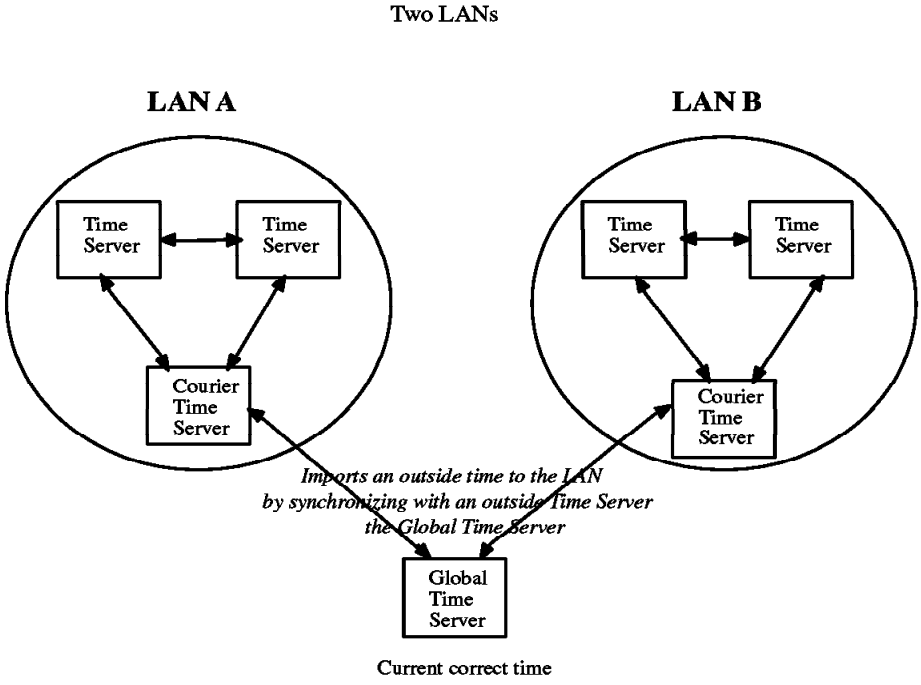


Figure 67. Courier Server

4.3.2.8 Time Provider

The notion of correct time must come from an outside source, the external Time Provider. This provides the time UTC, along with inaccuracy. It might be an atomic clock or some other time service, such as NTP (Network Time Protocol) of the Internet.

A time server synchronizes with a TP if one is available (otherwise it synchronizes with other time servers).

DTS supports the ability to interface with an External Time Provider through the Time Provider Interface. The External Time Provider itself is a hardware device, and is therefore outside the scope of DCE.

4.3.2.9 DTS Daemon

The DTS clerk and DTS server are both implemented as a *dtsd* process. A *dtsd* process becomes a server in response to a management command. Each of these processes communicate with other processes via authenticated RPC.

An administrator may issue commands to a *dtsd* via the *dtscp* command. A *dtsd* makes RPC call the other *dtsd*s configured as servers.

The *dtsd* command restarts the DTS daemon (clerk or server process).

When the host is re-booted, this command is automatically executed as part of overall DCE configuration procedure.

We might use *dtsd* interactively only when troubleshooting. To start the DTS daemon, we must use the *dce_config* shell command.

4.3.2.10 Distributed Time Service Control Program

The *dtscp* is a command-line interface that enables the administrator to synchronize, adjust, and maintain the system clocks in a distributed network. See Figure 68.

The control program commands can be used from within the control program or from the system prompt. From within the control program, just enter the *dtscp* command:

```
$ dtscp
dtscp>

 /* At this prompt we can enter any dtscp command.*/

dtscp> show current time

 To quit the dtscp prompt :

dtscp> exit
```

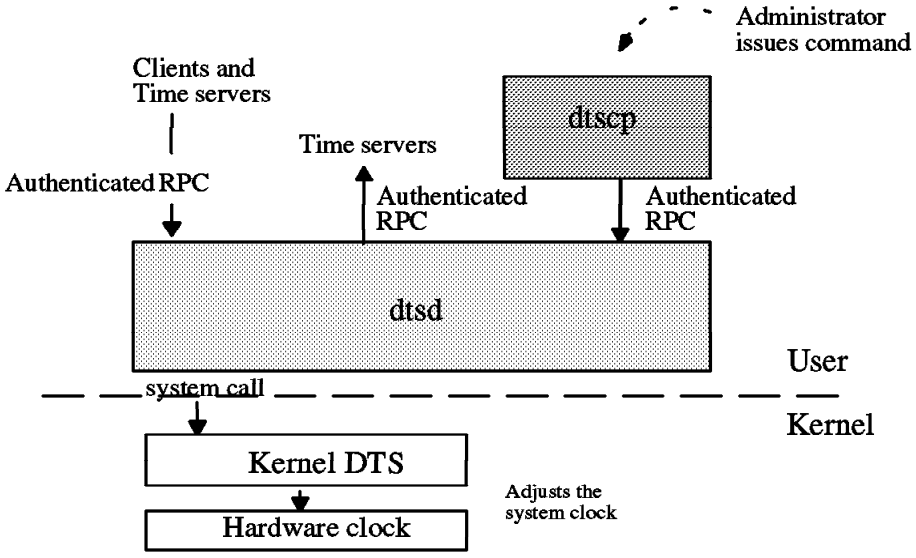


Figure 68. DTS Structure

4.3.2.11 Adjusting the Clock

When a client or a server has detected a difference between its local time and the *correct time* it has requested, usually it might want to adjust its local clock.

Under normal circumstances, such adjustment should never cause a clock to move backwards. We adjust the length of a clock tick. If a clock is too fast, the tick is adjusted so that when, the clock appears to have advanced by one second, 1001 milliseconds have actually transpired.

If the clock is too slow, the tick is adjusted so that when the clock appears to have advanced by one second, only 999 milliseconds have actually transpired. The clock is slowly adjusted until it shows the *correct* time.

By default, we adjust the clocks by one percent for a period long enough to effect the desired change.

If a clock is discovered to be wrong, 30 days wrong for example, adjusting the time gradually would take a long time. If a clock is off by more than a given value(ten minutes by default), then it is set immediately to the correct time rather than adjusting gradually. Such changes to a clock are normally done when the system is being re-booted.

The `synchronize dtscp` command causes the DTS entity to synchronize the clock in the system where the command is entered. The *set clock* argument specifies whether the clock is abruptly set or gradually adjusted to the computed *correct* time.

The `update time dtscp` command gradually adjusts the clock on the specified server node to the absolute time specified by the argument.

4.3.2.12 Notion of Epochs

To allow a new notion of time to be propagated throughout a cell, times are interchanged along with an epoch identifier (a simple integer).

A clerk or a server pays attention only to times within the receiver's epoch.

To introduce a *correct* time within a cell, a new epoch is created (by incrementing the epoch number). The machine that has received the new time then ignores times passed to it by other servers and vice versa.

An administrator then contacts each of the other machines in the cell and has them change to the new epoch. When a server is moved to another epoch, it does not advertise the time until it has received the current time within the new epoch from some other server.

The `change dtscp` command alters the epoch number and time on the local node. The epoch *integer* argument specifies the new epoch number, the time can be fixed by passing the `time` as *absolute-time* specifying a clock setting for the new epoch. If this is not specified, the server uses the current clock time with an unspecified inaccuracy and initiates a synchronization.

4.3.2.13 Time Representation

The Universal Time Coordinated (UTC) is the international time standard that has largely replaced the Greenwich Mean Time (GMT). This standard is administered by the International Time Bureau (BIH).

DTS uses opaque binary timestamps that represent UTC for all of its internal process. This timestamps cannot be read or disassembled. The DTS API allows applications to convert and manipulate them, but they cannot be displayed. DTS converts timestamps in ASCII text strings, which can be displayed.

Absolute Time Representation: An absolute time is a point on a time scale. DTS uses the UTC time scale. Absolute time measurements are derived from system clocks or external time-providers. DTS records the time in an opaque binary timestamp that also includes the inaccuracy and other information. Absolute Time is depicted on Figure 69.

When an absolute time is displayed, DTS converts the time to ASCII text, as shown in the following display :

```
1993-08-23-13:15:30.765-04:00I000.082
```

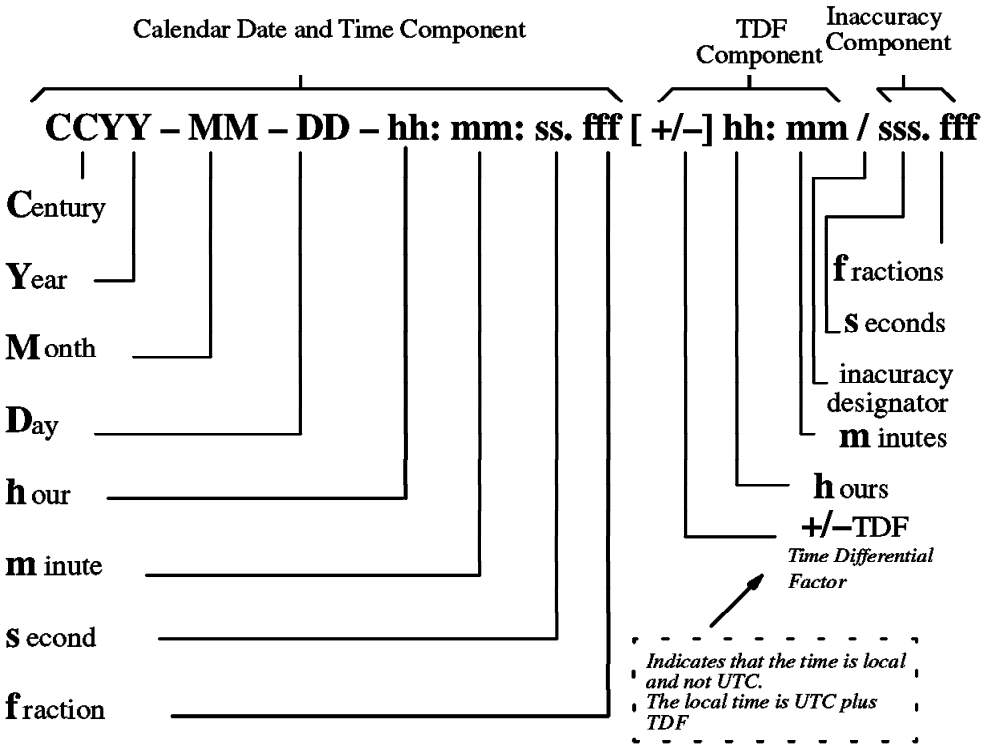


Figure 69. Absolute Time

DTS displays all times in a format that complies with the International Standards Organization (ISO) 8601 (1988).

Note: The inaccuracy portion of the time is not defined in the ISO standard, so time that does not contain an inaccuracy is accepted.

Variation to the ISO Format: Variations of the ISO format are also accepted as input for the ASCII conversion routines. In Figure 70 on page 106 the time (T) designator separates the calendar date from the time, a comma (,) separates seconds from fractional seconds, and the plus (+) or minus (-) character indicates the beginning of the inaccuracy component.

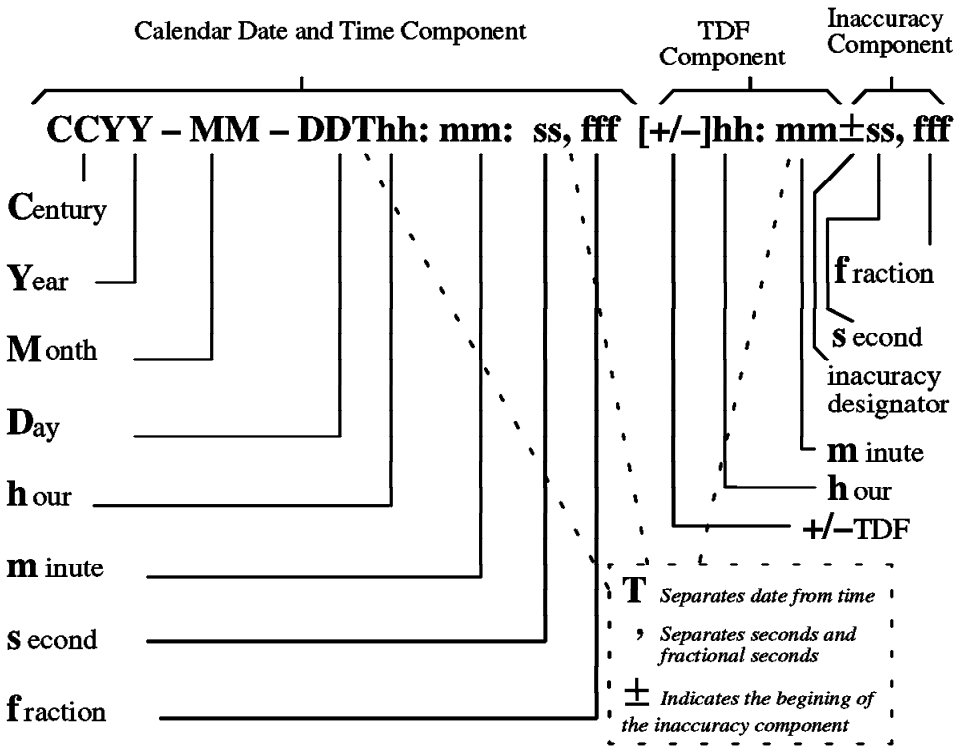


Figure 70. Variation of the ISO Time Format

Examples of Time Formats	
Represents	**Time Format**
1789-7-14-18:30:00	July 14, 1789 18:30 GMT without inaccuracy (default)
1789-7-14-12:01:00-05:00I100	Local time of 12:01 (17:01 GMT) on July 14, 1789 with a TDF of 5 hours and inaccuracy of 100 seconds
12:00 and T12	12:00 GMT in the current day, month, and year, with unspecified inaccuracy
1789-7-14	July 14, 1789 00,00 GMT with unspecified inaccuracy

Relative Time Representation: A relative time is a discrete time interval that is added to or subtracted from another time. A relative time is normally used as input for commands or system routines. See Figure 71 on page 107 for details about the structure of Relative Time.

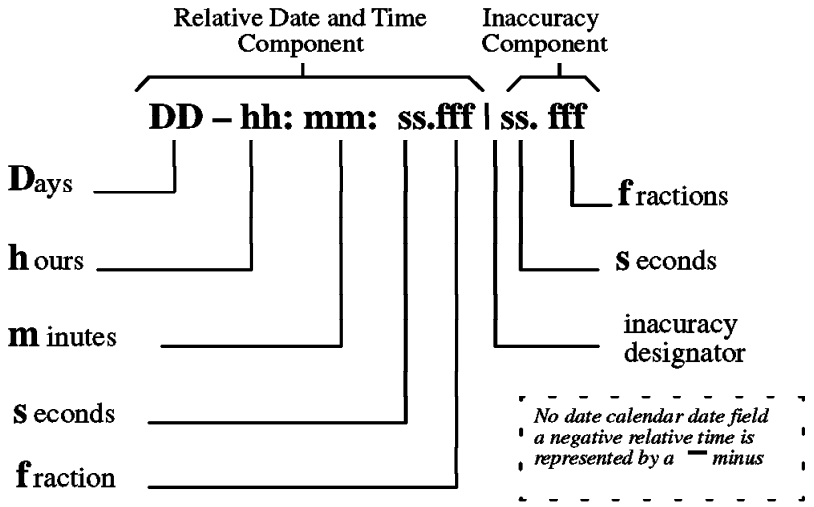
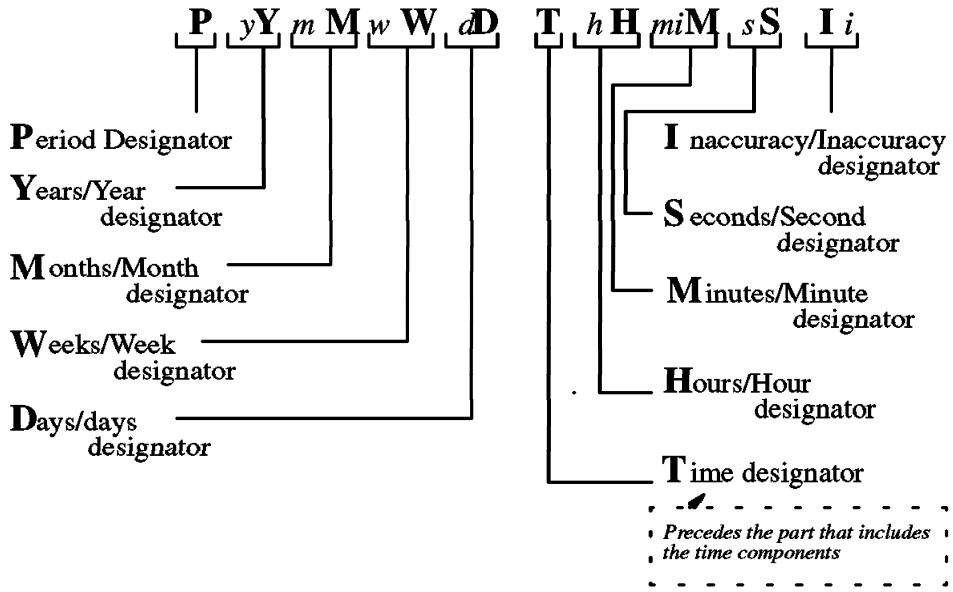


Figure 71. Relative Time Structure

Examples of Relative Time Formats	
Represents	**Time Format**
26-10:30:15.000I00.300	A relative time of 21 days, 10 hours, 30 minutes, and 15 seconds with an inaccuracy of 0.300 seconds
-20.2	A negative relative time of 20.2 seconds with unspecified inaccuracy (default)
10:15.1I4	A relative time of 10 minutes and 15.1 seconds with an inaccuracy of 4 seconds.

Period of Time: A period of time is represented by a data element of variable length, as depicted in Figure 72 on page 108.

Examples of Time Periods	
Represents	**Time Format**
P1Y3M15DT12H30M27S	A period of 1 year, 3 months, 15 days, 12 hours, 30 minutes, 27 seconds and unspecified inaccuracy
P3WI4	A period of 3 weeks and an inaccuracy of 4 seconds



y is the number of years preceding the designator Y
m is the number of months preceding the designator M
w is the number of weeks preceding the designator W
d is the number of days preceding the designator D
h is the number of hours preceding the designator H
mi is the number of minutes preceding the designator M
s is th enumber of seconds preceding the designator S
i is the number of seconds of inaccuracy preceding the designator I

Figure 72. Period of Time

4.3.2.14 DTS API

The DTS API is based on four structures:

- utc
- tm
- timespec
- reltimespec

utc Structure: DTS uses 128-bits binary numbers to represent time values internally. These binary numbers representing time values are referred to as binary timestamps.

The *utc* structure contains the following information :

- Count of 100-nanosecond units since 00:00:00:00, 15 October 1582 (date of the Gregorian reform to the Christian calendar)
- Count of 100-nanosecond units of accuracy applied to the preceding item
- Time Differential Factor (TDF), expressed as the signed quantity
- DTS version number

The API provides the necessary routines for converting between opaque binary timestamps and character strings that can be displayed and read by users.

tm Structure: The *tm* structure is based on the time in years, months, days, hours, minutes and seconds since 00:00:00 GMT, 1 January 1900.

This structure is defined in the *time.h* header file.

```
strut tm {
   int tm_sec;      /* Seconds (0-59)            */
   int tm_min;      /* Minutes (0-59)            */
   int tm_hour;     /* Hours   (0-23)            */
   int tm_mday;     /* Day of month  (1-31)      */
   int tm_mon;      /* Month  (0-11)             */
   int tm_year;     /* Year    -1900             */
   int tm_wday;     /* Day of week  (Sunday=0)   */
   int tm_yday;     /* Day of year  (0-364)      */
   int tm_isdst;    /* non zero if Daylight Saving Time */
}                   /* is in effect                     */
```

timespec Structure: The *timespec* structure is normally used in combination with or in place of the *tm* structure to provide finer resolution for binary times. The *timespec* structure provides the number of seconds and nanoseconds since the base time of 00:00:00 GMT, 1 January 1970.

This structure is in the *dce/utc.h* header file.

```
struct timespec {
  unsigned long tv_sec; /* seconds since of 00:00:00 GMT, 1 January 1970 */

  long tv_nsec;   /* additional nanoseconds since tv_sec */

} timespec_t;
```

reltimespec Structure: The *reltimespec* structure represents relative time. This structure is similar to the *timespec* structure, except that the first field is signed in the *reltimespec* structure.

This structure is in the *dce/utc.h* header file.

```
struct reltimespec {
  long tv_sec;    /* Seconds of relative time */
  long tv_nsec;   /* Additional nanoseconds of relative time */
} reltimespec_t;
```

DTS API Routines: Following are the DTS API routines by functional group

Time Retrieval Routines	
Description	**Routine**
utc_gettime	Returns the current system time and inaccuracy as an opaque timestamp
utc_getusertime	Returns the time and process-specific TDF, rather than the system specific TDF

Time Conversion Routines (tm structure)	
Description	**Routine**
utc_anytime	Converts a binary timestamp into a tm structure
utc_gmtime	Converts a binary timestamp into a tm structure that expresses GMT or the equivalent UTC

Time Conversion Routines (tm structure)	
Description	**Routine**
utc_localtime	Converts a binary timestamp into a tm structure that expresses local time
utc_mkanytime	Converts a tm structure and TDF into binary timestamp
utc_mkgmtime	Converts a tm structure that expresses GMT or UTC to a binary timestamp
utc_mklocaltime	Converts a tm structure that expresses local time to a binary timestamp
utc_mkreltime	Converts a tm structure that expresses relative time to a binary timestamp
utc_reltime	Converts a binary timestamp that expresses a relative time into a tm structure

Time Conversion Routines (timespec structure)	
Description	**Routine**
utc_binreltime	Converts a relative binary timestamp into two timespec structures that express relative time and inaccuracy
utc_bintime	Converts a binary timestamp into a timespec structure
utc_mkbinreltime	Converts a timespec structure expressing a relative time to a binary timestamp
utc_mkbintime	Converts a timespec structure into a binary timestamp

Time Conversion Routines (ASCII text)	
Description	**Routine**
utc_ascanytime	Converts a binary timestamp into an ASCII string that represents an arbitrary time zone
utc_ascgmtime	Converts a binary timestamp to an ASCII string that represents a GMT time
utc_asclocaltime	Converts a binary timestamp to an ASCII string that represents a local time
utc_ascreltime	Converts a binary timestamp that expresses a relative time to its ASCII representation
utc_mkasctime	Converts a NULL terminated character string, which represents an absolute timestamp, to a binary timestamp
utc_mkascreltime	Converts a NULL terminated character string, which represents a relative timestamp, to a binary timestamp

Time Manipulation Routines	
Description	**Routine**
utc_bountime	Given two binary timestamps, one before and one after an event, returns a single UTC time whose inaccuracy includes the event
utc_spantime	Given two (possibly unordered) timestamps, returns a single UTC time interval whose inaccuracy spans the two input timestamps
utc_pointtime	Converts a binary timestamp to three binary timestamps that represent the earliest, most likely, and the latest time

Time Comparison Routines	
Description	**Routine**
utc_cmpintervaltime	Compares two binary timestamps or two relative binary timestamps
utc_cmpmidtime	Compares two binary timestamps or two relative binary timestamps, ignoring inaccuracies

Time Calculation Routines	
Description	**Routine**
utc_abstime	Computes the absolute value of binary relative timestamp
utc_addtime	Computes the sum of two binary timestamps; the timestamps can be two relative times or a relative time and an absolute time.
utc_mulftime	Multiplies a relative binary timestamps by a floating-point value
utc_multime	Multiplies a relative binary timestamp by an integer factor
utc_subtime	Computes the difference between two binary timestamps, that expresses either an absolute time and a relative time, two relative times or two absolute times

Time Zone Information Routines	
Description	**Routine**
utc_anyzone	Gets the time zone label and offset from GMT by using the TDF contained in the input utc
utc_gmtzone	Gets the time zone label, given utc
utc_localzone	Gets the time zone label and offset from GMT, given utc

4.3.2.15 Time Provider Interface (TPI)

This section provides a brief overview of the Time Provider Interface for DCE DTS.

There are several devices that can acquire the UTC time values via radio, satellite, or telephone. These devices can provide standardized time values to computer systems. Normally one device is connected to a computer system; this device runs a process that interprets signals and translates them to time value. This time value can either be displayed or be provided to the server process running on the connected system.

To synchronize its system clock with UTC using such a device, the DTS server needs a software interface to this device to periodically obtain UTC. This interface serves as an intermediary between the DTS server and external time provider processes.

The DTS server requires the interface to obtain UTC time values and to determine the associated inaccuracy of each value. The interface between the DTS server process and the time provider process is called the *Time-Provider Interface* (TPI).

The external time provider is implemented as an independent process that communicates with a DTS server process through Remote Procedure Call (RPCs). Figure 73 on page 112 depicts RPC flow in the Time-Provider process.

- DTS daemon is the client

- Time-Provider process (TP process) is the server.

Both the RPC client (DTS daemon) and the server (TP process) must be running on the same system.

The TP process offers two procedures that DTS calls to obtain time values. These procedures are:

- ***ContactProvider***

  This procedure is the first called by DTS. It verifies that the TP process is running and obtains a control message that DTS uses for subsequent communications with the TP process and for synchronization after it receives the timestamps.

- ***ServerRequestProviderTime***

  After the TP process is successfully contacted, this procedure is called by DTS to obtain the timestamps from the external time-provider.

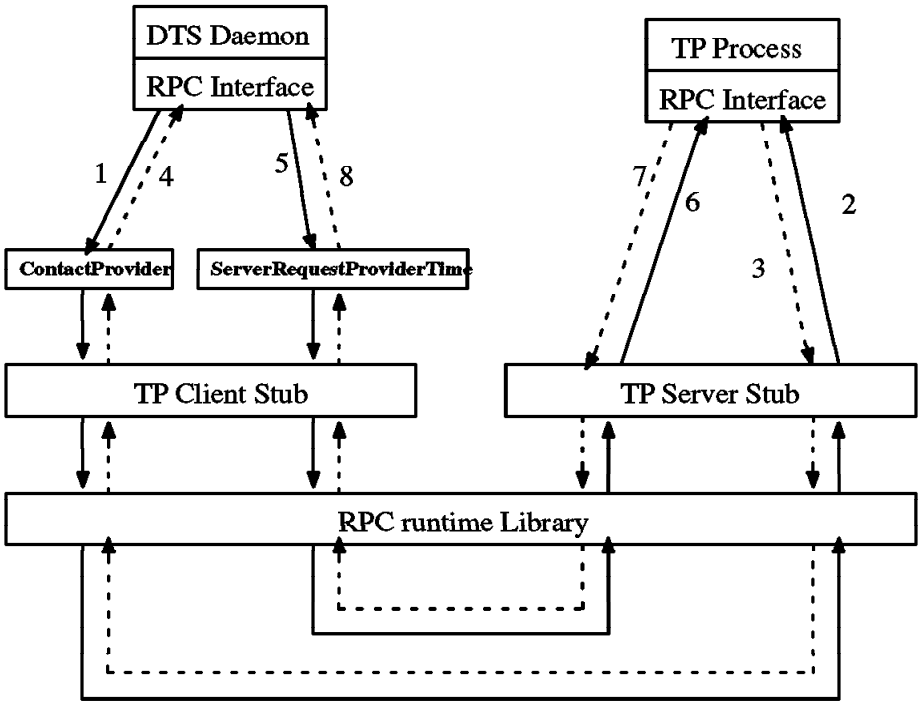


Figure 73. DTS Daemon and TP Process RPC Calling Sequence

Here are the steps of the remote procedure call from the DTS daemon to the TP process :

1. At synchronization time, DTS calls ***ContactProvider*** RPC.

   The input argument is passed to the TP client stub, dispatched to the RPC runtime library, and passed to the TP server stub.

2. The TP process receives the call and executes the ***ContactProvider*** procedure.

3. The procedure terminates and returns the results through the TP server stub, the RPC runtime, and the TP client stub.
4. The procedure terminates in the DTS call, where the returned parameters are examined.
5. The DTS calls the ***ServerRequesterProviderTime*** remote procedure.

   The input parameters are passed to the client stub, dispatched to the RPC runtime and passed to the TP server stub.
6. The TP process receives the call and executes the ***SeverRequestProviderTime*** procedure.
7. The procedure terminates and returns the results through the TP server stub, the RPC runtime and the TP client stub.
8. The timestamps are returned as output parameters.

   DTS then synchronizes using the timestamps returned by the external time-provider.

For more information on the TPI, see the *OSF DCE Application Development Guide*.

Part 2. Solutions

Chapter 5. Directory, Naming and Time Solutions

This chapter gives an overview of the real products implementing the directory architectures introduced in Chapter 2, “Directory Technologies and Standards” on page 9, Chapter 3, “Naming Technologies and Standards” on page 73, and Chapter 4, “Time Services Technologies” on page 95. It comprises two sections, one for Directory and Naming, and one for Time.

5.1 Directory and Naming Implementations

Table 8 lists the standard operating platforms and indicates the directory technologies available for them. Note that in many cases you must install or buy additional function in order to implement a particular directory technology. Most of the X.500 directory implementations available are part of their implementations of OSF/DCE, while most of the DNS implementations are available as part of TCP/IP. APPN consists of a complex set of functions, not all of which are available on every version of every platform. Please consult the APPN literature for detailed information about exactly which functions are available on which platforms.

Note that in addition to the platforms mentioned below, nearly every platform has some form of TCP/IP available. At the end of this section, there is a list of platforms that also have implemented or announced some form of DCE.

Table 8. Directory and Naming Technologies supported by Various Platforms

Product	X.500	DNS	APPN	Notes:
MSDOS/PCDOS		TCP/IP		
Windows**		TCP/IP, DCE Cell Directory Services		
OS/2	DCE for OS/2	TCP/IP	OS/2 V2 CM/2 V1.1	(1)
AIX/6000	AIX DCE	TCP/IP	AIX SNA Server/6000	
AS/400	AIX DCE	TCP/IP	Integrated in OS/400*	
MVS	MVS OpenEdition DCE	TCP/IP	ACF/VTAM	
VM		TCP/IP	ACF/VTAM	
UNIX		TCP/IP		
Novell** Netware** 3.1		TCP/IP		
Novell** Netware** 4.0	Network Directory Services	TCP/IP		(2)
Banyan Vines**	Streettalk**			(3)

Notes.

1. APPN support was already available with Networking Systems/2, an extension of the OS/2 V1.3 Communications Manager support.
2. With Netware 4.0, Novell introduced an X.500 based directory service, called the Netware Directory Services. Table 9 summarizes the main features of NDS.

Table 9. Summary of Novell Netware Directory Services (Source: LAN Systems Inc.)

Feature	**Implementation**
Development Environment	Client-server, C Language
Location of Data	Distributed and replicated
Network Operating System	Netware 4.0, Native Netware for OS/2, Netware for Unix (20+ OEM versions)
Client operating systems	DOS, Windows, OS/2, Macintosh, Unix
System memory requirements	DOS client—60 kb (can be reduced using memory management) Server—8 Mb
Security	Authentication via RSA public key-based multilevel network-wide access control
Default object types	Alias, bindery object, computer, country, directory map, group, locality, NCP server, organization, organizational role, organizational unit, print server, printer, profile, queue, storage management services, top, transaction set, unknown user, volume
Naming levels	Unlimited levels
Standards	Based on X.500; initial X.500 compatibility via gateway; committed to full X.500 implementation

3. Like Novell, Banyan has implemented a variation on the X.500 standard, which they call Banyan Streettalk**. Table 10 summarizes the main features of Banyan Streettalk.

Table 10 (Page 1 of 2). Summary of Banyan Streettalk (Source: LAN Systems Inc.)

Feature	**Implementation**
Development Environment	Client-server, C Language
Location of Data	Distributed database with read-only lookup for Streettalk Directory Assistance
Network Operating System	All Vines** versions, SCO** Unix, Netware (end of 1992)
Client operating systems	DOS, Windows, OS/2, Macintosh
System memory requirements	DOS client—122 kb (can be reduced using memory management) Server—8 Mb
Security	Authentication via proprietary token-based Vines security services
Default object types	Users, nicknames, lists, services, servers, groups

Table 10 (Page 2 of 2). Summary of Banyan Streettalk (Source: LAN Systems Inc.)	
Feature	**Implementation**
Naming levels	Three levels
Standards	X.500-like; committed to full X.500 interoperability

In addition to the main platforms listed above, the following platforms have also announced or are delivering some form of DCE implementation.

- Atrium –

  Distributed Common System Services (DCSS) kit available on SPARCstations**,
  HP** 9000, RISC System/6000 DACM, Notifications, SQL Query Unix, Macintosh and PCs DAZEL: available now.

- Bull – GCOS - (future)
- BOS2 for DPX/2 from Bull
- PC/DCE for Zenith Data Systems
- Cray Research – UNICOS
- DEC – OpenVMS(VAX**) & AXP), Ultrix**, and OSF/1**(AXP & MIPS**)
- Data General – DCE for AViiON** System
- Ellery Systems – Ellery Open Systems** available on HP 9000, SPARC**, DEC 3100/5000/Alpha**, Intel**
- Gradient – Windows 3.x & SVR4 for PCs
- Hewlett-Packard** – MPE/iX** & HP-UX**
- Hitachi** – DCE on OSF/1, HI-UX (UNIX) and DCE VOS3 (mainframe OS)
- Kapsch – VM/CMS
- NCR** – NCR DCE R1.00 (executive runtime) entire family of System 3000 Open Cooperative Come Env (OCCE)
- DOS/Windows 3.x Zenith Data Systems PC from Bull
- NEC** – SVR4
- Olivetti** – DCE Development Toolkit available mid 1993 LSX 5000 Open System Architecture (OSA)
- Oracle** – ORACLE7
- SCO** – SCO/UNIX: DCE Developer's Version
- Siemens Nixdorf** – DCE SINIX V1.0 for UNIX beta

  DCE SVR4 Reference Port
  DCE WIN for Windows Windows clients
  DCE for mainframes: BS 2000 mainframe operating system
  PC-DCE announced: WX200; MX300 (Intel) Gradient Technologies, Inc

- Stratus** – DCE/SVR4 on FTX; SVR4 on Stratus Systems
- SYBASE** – SYSBASE Open Client Library System
- Tandem** – Guardian/POSIX & SVR4
- Transarc – Encina** for Solaris** 1.1

OLTP for IBM, Hitachi, HP, NEC
Encina for RISC System/600
Encina for Solaris 2.2

- USL – SVR4.2

  DCE Secure Core
  SVR4 on Intel for SVR4 OEM
  DCE Global Directory Service
  Distributed File System

- Microsoft – Windows/NT will have interoperability with DCE-RPC

5.2 Time Service Implementations

Table 11. Time Service Technologies supported by Various Platforms

Product	**DCEdts**	**Other Distributed Time Provider**	**Notes:**
MSDOS/PCDOS			
Windows	DCE for Windows Client		
OS/2	DCE for OS/2		
AIX/6000	AIX DCE		
AS/400	AIX DCE		
MVS	MVS OpenEdition DCE	Sysplex Timer	(1)
VM			
UNIX	(In individual DCE implementation)		
Novell** Netware** 4.0	Time Synchronization Services (TSS)		
Banyan Vines**	??		

Notes:

1. The 9037 Sysplex Timer provides common time for all MVS systems connected in a sysplex. It's a matter of debate as to whether a sysplex constitutes a distributed system (it's certainly not heterogeneous), but there are indications that the role of the sysplex will be expanded in the future.

Part 3. Appendixes

Appendix A. OSI Reference Model

The international standards most relevant to open systems are those that support the Open Systems Interconnection (OSI) reference model. The OSI reference model is described in International Standard ISO 7489-1, published in 1984. The OSI reference model has become the basis for a number of international standards addressing electronic communications. Examples include the X.25 packet-level protocol, the ISO 8326 standard for basic connection-oriented session service, and the ISO 8571 standard for file transfer, access, and management (FTAM).

The OSI reference model is important for several reasons: because of its acceptance in the standards community as a concept, because of the number of other standards that have been written to support it, and because as IBM has been supporting international standards as a means of realizing open systems, many of these standards have been selected from the OSI family of standards.

This model conceptually depicts the communications process between two computer systems as two “protocol stacks.” Each end-user or application process resides at the top of a stack. Each stack is then divided into seven layers. Separate communications protocols can be developed for each layer, thus subdividing the overall task of communications into distinct subtasks. An analogy exists with IBM′s Systems Network Architecture (SNA), which also separates the communications task into subtasks. The SNA model also uses seven layers, but they are not identical with the OSI layers.

From the bottom up, the OSI reference model consists of the physical layer, the data link layer, the network layer, the transport layer, the session layer, the presentation layer, and the application layer.

A.1.1.1 Physical Layer

The physical layer basically supports the electrical interface between pieces of equipment. It is more oriented toward computer hardware than toward computer software. The purpose of the physical layer is to move bits between a sending device and a receiving device, where the receiving device may be an intermediate system or an end system. This layer is responsible to activate, maintain, and deactivate physical connections for the transmission of bits. It does not specify the medium, which can include cabling, such as twisted pair, coaxial, and fiber optic.

Physical layer standards are usually found in CCITT Recommendations in the V, X, and I series. The V series is for analog modems; the X series is for digital data networks; and the I series is for Integrated Services Digital Network (ISDN).

A.1.1.2 Data Link Layer

The data link layer supports transmission of data over individual data links. A link is considered to exist between any two directly connected computers, and may include one or more physical connections. In the OSI reference model, each layer adds function to the services provided by the adjacent lower layer. The additional data link layer function includes sequence control, flow control, error detection, recovery, and notification, and the ability to manage multiple physical layer connections. An example of a data link layer protocol is the High-level Data Link Control (HDLC) protocol, which is specified in ISO 3309, 4335, and 7809.

A.1.1.3 Network Layer

The network layer supports transmission of messages or packets between source nodes and destination nodes. The source and destination nodes represent endpoints in the movement of messages through an open system, and may include one or more data links and one or more physical connections. The service definition for the network layer is found in CCITT X.213 and ISO 8348. In addition to the services already provided by the data link layer and the physical layer, the network layer provides receipt confirmation, expedited data, quality of service, and user data. The network layer hides many details of data transmission, so that end systems sending information only need to specify the final destination and desired quality of service.

A.1.1.4 Transport Layer

The transport layer supports transmission of messages between end users. The transport layer has the purpose of delivering message packets with end-to-end integrity, which means that data packets are not lost, duplicated, delivered out of order, or damaged in delivery. The transport layer builds upon the layers below it, and if the lower layers provide sufficient services, the transport layer may only need to pass through services from the network or data link layers.

OSI defines five classes of transport service, TP0 through TP4, which vary depending on the intended application and the kind of network service provided. For example, class TP0 is intended to support delivery of X.400 electronic mail.

A.1.1.5 Session Layer

The physical, data link, network, and transport layers are sometimes referred to as the *OSI lower layers* because their combined purpose is to provide reliable, robust delivery of information between end systems in open networks. The remaining OSI layers— session, presentation, and application— are sometimes referred to as the *OSI upper layers* because their focus is different. They assume the presence of a reliable end-to-end communication subsystem below them, and then set the objective of providing application-oriented services to support end users of computers in open systems.

The session layer supports communications sessions between end users.

A.1.1.6 Presentation Layer

Historically, the presentation layer was intended to add function to support transmission of information necessary to *present* data correctly to a receiving end system— for example, information about character sets and end-of-line markers. This purpose evolved into support for the abstract representation of transmitted data. Because of unpredictable internal data representations in end systems in open networks, the presentation layer supports transmission of data in a standard intermediate form, in which data structures are defined in *Abstract Syntax Notation One (ASN.1)* with the *Basic Encoding Rules (BER)* as the transfer syntax. ASN.1 is described in Appendix C, “Overview of ASN.1” on page 135.

Presentation layer functionality is described in ISO 8822 and ISO 8823.

A.1.1.7 Application Layer

The application layer supports protocols and services required by particular user-designed application processes. functions satisfying particular end-user requirements are contained in this layer, whereas representation and transfer of information necessary to communicate between applications are the responsibility of the lower layers.

Application service elements (ASEs) are the building blocks of the application layer. Typically, a user application is referred to as an *application process*, which for communications purposes interacts with the OSI environment through a *user element*. The user element, together with a set of application service elements, constitutes an *application entity*.

Application service elements may provide a general service usable by various user applications, or they may be designed to support specific user applications. They are referred to as *common application service elements (CASEs)* or *specific application service elements (SASEs)* respectively.

An example of a CASE is the Association Control Service Element (ACSE), which provides services to establish and delete associations between application entities. It is a common application service element that typically supports other application-layer standards.

File transfer, Access, and Management (FTAM) and Message Handling System (MHS) are examples of SASEs— application service elements that support specific application-oriented tasks for end users.

Appendix B. Time in MVS

Time recording and maintenance are very important to any MVS installation. The requirement for accurate time becomes more important when several systems are connected together in a sysplex, because in many ways the systems in a sysplex act as a single system. The IBM 9037 Sysplex Timer was introduced to provide this synchronization. Time synchronization between distant systems or sysplexes can also be important.

A multisystem XCF complex (multisystem sysplex) requires that the clocks of all of the systems be very closely synchronized (typically to a very few microseconds) to maintain data integrity across the systems. This can be accomplished in two ways. If all MVS systems in the sysplex are in the same CPC (either as logical partitions or under VM), clock synchronization is guaranteed because all systems use the same TOD clock. If the MVS systems are spread across more than one CPC, a separate external clock, called the Sysplex Timer, is required to synchronize the TOD clocks of the individual CPCs. This service is provided by the IBM 9037 Sysplex Timer, which connects to the individual CPCs by fiber optic links similar to those used by ESCON channels. *Planning for the 9037 Sysplex Timer* provides detailed information about the IBM 9037 Sysplex Timer.

This appendix discusses some concepts of timing in an MVS environment and how they are handled by the Sysplex Timer and MVS/ESA Version 4. The following main topics are discussed:

- Time and Date Concepts
- Sysplex Timer, TOD clock, and MVS/ESA Clock Concepts
- Time adjustments.

B.1 Time and Date Concepts

This section provides a brief history of how changes to the calendar were made to compensate for the earth's rotational differences and how "universal" timekeeping evolved. While it may appear that this has little bearing on timekeeping in MVS, in fact it introduces some concepts that are required when trying to understand how the Sysplex Timer, the CPC TOD clock, and MVS interact. It discusses:

- Why time and date corrections are necessary
- Leap year and leap second corrections
- Atomic Time and Universal Time Coordinated

After understanding why time and date corrections occur, you can understand why accurate time synchronization is necessary, even on a world-wide basis, to maintain chronological integrity. Also, after understanding the "time and date" terms, you can determine how they relate to Sysplex Timer and MVS operation.

B.1.1 Why Time and Date Corrections Are Necessary

The earth does not revolve around the sun either in an integral number of days or at a constant rate. This results in a requirement for leap years and, more recently, leap seconds. Calendars were invented before people knew about these variances. To provide a brief historical background, there were two occasions when the leap year differences became so great compared to the solar day that it forced corrections to the yearly calendar.

B.1.1.1 Leap Years and Their Correction

The first leap year correction (46 BC) added 80 days to adjust for seasonal (equinox) differences. This was understandable because not only was there disparity in the calendars used, but people were just beginning to understand that rotational variances existed. At that point, the year was divided into 12 equal months, leap year compensation began, and one calendar system became widely accepted.

By 1582, it was apparent that the existing leap year compensation was not completely accurate, so October 4th was followed by October 15th to again synchronize the date/equinox difference. Coincidentally, both for that correction and for leap year corrections, the next day of the week does not change (for example, Friday always follows Thursday). Since the 1582 calendar correction, a leap year occurs if the year is divisible by four, but if it is divisible by 100, it must also be divisible by 400. Therefore, 2000 is a leap year but 1900 and 2100 are not.

B.1.1.2 Leap Seconds and Their Correction

Leap years compensated for gross inaccuracies, but because keeping “exact” time became more necessary, sophisticated methods were developed to compensate for smaller time differences. For example, because of eccentricity in the earth′s orbit, days are longer in January than in July. Also, the earth′s rotational rate varies slightly and irregularly. This evolved into the requirement to also correct for leap seconds.

B.1.2 Atomic Time

The atomic clock, constructed in the mid-1950s, caused an awareness of these minor differences and allowed for compensation. By defining the length of an “average” day and year, they could be matched to the motion of the earth around the sun. Because Atomic Time advances consistently compared to “rotational time,” sunrise and sunset would not always occur at the same time on each date every year. To make sure that this always happens, leap seconds must be used to match Atomic Time to the earth′s rotation.

B.1.3 Universal Time Coordinated (UTC)

Before 1972, in the United States clocks were adjusted forward or backward in 100-millisecond steps to stay synchronized with rotational differences. Other nations, however, made their own adjustments, thereby causing world-wide differences and problems similar in degree to what previously occurred because of leap-year discrepancies.

Many current activities require synchronization to sub-second levels, but it is not practical to vary clock rates to match the earth′s rotation. Therefore, once it was determined that clock rates should remain constant, small adjustments could be made when there were variations between Atomic Time and rotational time.

The concept of coordinating time universally was introduced in 1972 to provide world-wide time consistency. Most countries use Universal Time Coordinated (UTC) because today's requirements for time accuracy are greater than ever before. Many countries based their time standards on Greenwich Mean Time (GMT), an older time standard than UTC. For all practical purposes, UTC and GMT can be considered equivalent. UTC and GMT are periodically adjusted by one or two leap seconds per year. This adjustment can be simply expressed as:

```
UTC = Atomic Time ± number of leap seconds
```

Therefore, leap-second compensation must occur in one-second increments that can be added to, or subtracted from, Atomic Time to compensate for the yearly rotational differences. If the earth rotates:

- Faster than average, a leap second is subtracted
- Slower than average, a leap second is added.

This allows uniform time to occur for at least six months. Because the rotational variation cannot be predicted early, however, the leap second value can be determined (at best) only two months ahead.

With the introduction of the 9037, the processor time-of-day (TOD) clock is synchronized with Atomic Time rather than UTC. To correctly set the TOD clock, the number of leap seconds that have occurred must be entered into the 9037. Leap seconds normally occur at the end of the day (UTC time) on June 30 or December 31. The following shows the time in 24-hour format at the end of a normal day, and for a leap second:

A normal day:
23:59:58, 23:59:59, 00:00:00, ...

A positive leap second:
23:59:58, 23:59:59, 23:59:59, 00:00:00, ...

A negative leap second:
23:59:58, 00:00:00, ...

As of December 31, 1989, 15 positive leap seconds had been inserted. Positive leap seconds, beginning at 23:59:60 UTC and ending at 00:00:00 UTC, were inserted in the UTC timescale on 30 June 1972, 31 December 1972-1979, 30 June 1985, 31 December 1987, and 31 December 1989.

B.2 Sysplex Timer, TOD clock, and MVS/ESA Clock Concepts

Before MVS/ESA Version 4, MVS was concerned with only two times, GMT and local time. These two times differed by a fixed amount, which was usually subject to change only twice a year, when going from standard time to summer time and back again. Starting with MVS/ESA Version 4 Release 1, MVS can be dealing with up to three distinct representations of time:

- Atomic Time or Absolute Time
- Universal Time Coordinated (UTC) or Greenwich Mean Time (GMT), which is the Atomic Time plus or minus the leap seconds accumulated since 1972
- Local Time, which is UTC plus or minus a fixed offset.

Of the three time representations, only absolute time offers consistent, monotonically increasing time values. Both UTC and local time occasionally skip or repeat time values. For this reason, absolute time is the best choice for applications that have to unambiguously timestamp their data (as in logging).

The Sysplex Timer synchronizes, to a very close tolerance, the TOD clocks of all the CPCs to which it is attached. The precision of this synchronization is to a certain extent dependent upon the CPC hardware, but TOD clocks in the CPCs typically vary from one another by only a very few microseconds at most.

The Sysplex Timer periodically queries the TOD clocks. If the time difference between the TOD and the Sysplex Timer exceeds an internal threshold, the Sysplex Timer immediately adjusts the TOD clock to the correct time.

Even though the user can enter any initial time into the Sysplex Timer he wishes, the Sysplex Timer treats that time as absolute or atomic time. This is the time to which the physical TOD of the CPC is synchronized.

In addition to the absolute time, the Sysplex Timer also provides offset information for leap seconds and local time to the CPC. The offset information can be used by an operating system running in the CPC to calculate the correct values for UTC and local time. The Sysplex Timer can be programmed to change the offset information at a predetermined time. You can specify that a leap second is to be added or subtracted at midnight on December 31, for example, or that the local time offset should change from +1 hour to +2 hours at 1 AM on April 6, 1994. When those events occur, the Sysplex Timer transmits the changed offsets to the CPC. The CPC then interrupts the operating systems running on it to inform them of the offset changes. Operating systems that are suitably programmed (such as MVS/ESA Version 4) can then automatically adjust their own offsets to reflect the change. Thus, it is not necessary to issue a SET TIME command or to IPL the system to change from standard time to daylight savings time, for example.

Note that while all these adjustments are taking place, no adjustment is made to the TOD clock or to the “absolute” time running in the Sysplex Timer. Adjustments to the TOD clock are made when it is determined to be out of sync with the Sysplex Timer. The time in the Sysplex Timer is changed when an external event changes its setting. This event can be a manual change to the Sysplex Timer setting or one that is caused by an out-of-sync condition between the Sysplex Timer and its external time source.

B.3 MVS/ESA Time

As mentioned above, MVS/ESA Version 4 keeps up to three separate representations of time:

- Absolute Time
- GMT Time (UTC)
- Local Time.

B.3.1 Time Representations and Guidelines

If you do not specify ETRMODE YES, Time of Day (TOD) representations in MVS are the same as in pre-Version 4 releases. If you do specify ETRMODE YES, MVS can display up to three distinct “times” at any particular moment. The following guidelines show which time you should use for a particular purpose.

Timestamps fall into three categories:

- The most obvious use is to obtain the correct date and time of day. The date and time may be either local time and date or Greenwich Mean Time (GMT) and date.
- Timestamps are frequently used as a mechanism to ensure the correct sequencing of entries in logs, data records, or recovery records.
- TOD clock values are used to calculate the time interval between two events.

The use for which a TOD clock value is gathered determines which methods can be used to obtain the TOD value. Note that the method required for one use is not necessarily appropriate for another use. It is especially dangerous to attempt to obtain time of day values by issuing the STCK instruction. If you have specified ETRMODE YES, the TOD clock value returned is NOT necessarily GMT. If you have specified a non-zero leap second offset in the Sysplex Timer, the TOD value represents an “absolute time” value instead.

Table 12 lists several uses for timestamp values and the ways to obtain them from MVS. The TIME macro is the recommended method to obtain times of day (local or GMT), since the STCK could deliver absolute time:

Table 12. Timestamps in MVS

Use for clock value	STCK *	STCK+LTO *	TIME STCK	TIME LT	TIME GMT
Local time of day	No	No	No	Yes	No
GMT time of day	No	No	No	No	Yes
To sequence log entries	Yes	No	Yes	No	No
To order recovery info	Yes	No	Yes	No	No
To calculate an interval	Yes	No	Yes	No	No

Note: The last three columns are TIME macro parameters.

where:

***** The STCK instruction or the STCKSYNC macro.

LTO The local time offset value. The whole doubleword value is contained in CVTLDTO. CVTTZ contains only the high word, so if it is used, the resulting value is accurate only within approximately one second.

B.3.2 Time Compatibility between MVS Releases

MVS/ESA Version 4 is able to distinguish between absolute time (time returned by STCK) and GMT time (absolute time adjusted by leap seconds) when a Sysplex Timer is attached to the CPC. In older releases of MVS, the value returned by STCK and GMT are equivalent. Therefore, systems that are controlled by a single Sysplex Timer could show different GMT and local time values depending on their release levels and CLOCKxx parameter values. The values returned by STCK (the TOD value) should be the same for all systems, however.

B.4 Time Adjustments

With MVS/ESA Version 4, it becomes more important to understand how the various time representations might be adjusted to keep the values in sync. This is also important because it helps explain exactly the implications of the ETRDELTA specification in the CLOCKxx parmlib member. There are basically three places where out-of-sync conditions can occur and adjustments must be made:

- Between the Sysplex Timer and the CPC TOD
- Between an external time source and the Sysplex Timer
- "Planned" changes to the offsets for leap seconds or change from winter to summer and back again.

Each of these changes is handled slightly differently, and each has different implications.

B.4.1 Adjustments to the TOD Clock

When a Sysplex Timer is attached to a CPC, the Sysplex Timer ensures that the two TOD clocks (Sysplex Timer and CPC) stay in close synchronization by testing the time in the CPC at frequent intervals. If the two clocks differ by more than a certain amount, the Sysplex Timer raises a synchronization check and causes the CPC TOD clock to be set to the correct time. The amount of divergence that causes the synchronization check depends upon the hardware, but is on the order of a very few microseconds.

If the difference does not exceed the value of ETRDELTA, MVS is not aware of the change, although a time adjustment that causes the CPC TOD to repeat time values causes MVS to stop processing until the TOD starts showing new time values. This guarantees that there can be no out-of-sequence timestamping in MVS due to a TOD adjustment.

If the difference exceeds the value of ETRDELTA, MVS is notified. MVS then puts all of the members of the sysplex into a restartable wait state.

An ETRDELTA value of 10 seconds (the default) is likely to be exceeded in only two cases: if there is a hardware error in either the Sysplex Timer or the CPC TOD clock, or if someone manually sets the Sysplex Timer to a different value.

B.4.2 Adjustments From an External Time Source

It is possible to use an external time source (such as a radio transmitter or telephone connection) to ensure that the Sysplex Timer is at the correct time. This synchronization may be important for enterprises that have widely scattered sites whose times have to match closely.

The external time source provides UTC or GMT times and the Sysplex Timer runs on absolute time. If the Sysplex Timer is set up with a leap second offset, the Sysplex Timer is set to a time that is the Universal Time Coordinated offset by the number of leap seconds.

While the Sysplex Timer and the CPC TOD are synchronized many times per second, synchronization between the Sysplex Timer and the external time source typically occurs much less often. In fact, there is no set synchronization period for these events. They are totally installation dependent and could occur

days apart. Thus, it is possible that the time kept in the Sysplex Timer could differ significantly from the external time source.

In order to avoid disruption due to sudden large changes in the TOD clock setting (possibly even exceeding ETRDELTA), time differences between the Sysplex Timer and the external time source are compensated very gradually. A time difference of one second could take as long as 24 hours to be adjusted, for example. This avoids the possibility that log records and timing events display anomalies or that ETRDELTA is exceeded in case of a timing adjustment due to a mismatch between the external time source and the Sysplex Timer.

Because the external time source uses UTC, leap seconds have to be accounted for. If they are not, the Sysplex Timer has to adjust itself each time a leap second is added to or subtracted from UTC.

For example, assume that a leap second is added to UTC at midnight on December 31, 1991. The time coming from the external timer would then appear as follows:

23:59:58, 23:59:59, 23:59:60, 00:00:00, 00:00:01

If you chose not to compensate for leap seconds (left the offset at zero), the Sysplex Timer would assume the following UTC times over the same period:

23:59:58, 23:59:59, 00:00:00, 00:00:01, 00:00:02

Any attempt to synchronize the two after midnight would show a difference of one second, and the Sysplex Timer would be slowed down slightly until it was once again in sync with the external time source.

By adding an additional leap second to the leap second offset in the Sysplex Timer (to take effect at midnight), there would be no need to slow down the Sysplex Timer because there would be no discrepancy.

B.4.3 Planned Adjustments to Time Offsets

As explained earlier, the Sysplex Timer should not show discontinuities in time; that is, no time should be skipped and no time should be repeated. Such discontinuities should only occur due to a malfunction or manual intervention (if someone were to reset the time). In real life, there are time discontinuities, however, namely changes due to leap seconds and seasons.

In order to accommodate these, the Sysplex Timer maintains offsets which are added to or subtracted from the “absolute” time of the timer clock. Thus, a season or leap second change results in a change to the offsets and not to the TOD time. Therefore, season or leap second changes do not result in an event which would cause ETRDELTA to be compared. These offsets are communicated to MVS, which uses them to produce the times listed in Table 12.

Appendix C. Overview of ASN.1

Within the framework of the ISO standards supporting the OSI reference model, ASN.1 serves as a standard language for specifying data types and values. The following section provides an overview of ASN.1.

C.1.1 Background

ASN.1 (Abstract Syntax Notation One) is a formal notation for documenting abstract data types.

- ASN.1 is defined in ISO 8824 and CCITT X.208.
- The Basic Encoding Rules (BER) for ASN.1 are defined in ISO 8825 and CCITT X.209.

C.1.2 Purpose

To describe data types in a universal, abstract, and unambiguous manner in order to facilitate data communications between heterogeneous systems. ASN.1 lets you *define data types* and *specify values* for data types.

C.1.3 Selected Terminology

- *Abstract syntax* is a data representation independent of programming language, operating system, or machine architecture.
- *Encoding rules*, such as the BER, are applied to abstract syntax. This yields *transfer* or *concrete syntax* that is used when data is actually transferred between systems.
- *Tags* identify data types.
- *Type reference* is a name associated with a data type, beginning with an uppercase letter.
- *Value reference* is a name associated with a specific value, beginning with a lowercase letter.
- *Productions* are expressions in which names are associated with collections of sequences of ASN.1 characters.
- An *item* is a sequence of ASN.1 characters, separated from other sequences by a space or new line.
- A *simple type* is defined by referring to other types.
- A *subtype* is a type whose values are a subset of the values of another type. You can use it to create a range of values.

C.2 ASN.1 Structural Elements

The structural elements of ASN.1 are the elements that are used in the documentation of abstract data types.

C.2.1 ASN.1 Character Set

- A to Z
- a to z
- 0 to 9
- : = , { } < >] .
- ()[]-′″

C.2.2 Reserved Words

- ABSENT
- ANY
- APPLICATION
- BEGIN
- BIT
- BOOLEAN
- BY
- CHOICE
- COMPONENT
- COMPONENTS
- DEFAULT
- DEFINED
- DEFINITIONS
- END
- ENUMERATED
- EXPLICIT
- EXPORTS
- EXTERNAL
- FALSE
- FROM
- IDENTIFIER
- IMPLICIT
- IMPORTS
- INCLUDES
- INTEGER
- MAX
- MIN
- MINUS-INFINITY
- NULL
- OBJECT
- OCTET
- OF
- OPTIONAL
- PLUS-INFINITY
- PRESENT
- PRIVATE
- REAL
- SEQUENCE
- SET
- SIZE
- STRING
- TRUE
- TAGS
- UNIVERSAL

- WITH

C.2.3 Productions

ASN.1 productions have three parts:

- Name
- ::= (which means *defined as*)
- One or more named collections of sequences of ASN.1 characters, which can be separated by] (which means *or*).

C.2.4 Example of ASN.1 Production

RecordKey ::= INTEGER
firstKey RecordKey ::= 1
(RecordKey is a type reference; INTEGER is an ASN.1 data type; firstKey is a value reference.)

C.2.5 Miscellaneous Rules

- ASN.1 is case sensitive.
- There is no restriction on line length.
- A comment begins with double hyphens and ends with double hyphens or at end of line.
- Recursive productions are allowed.
- Leading zeros are not allowed and there are no spaces within items.
- Empty items are allowed.

C.2.6 ASN.1 Data Types

Builtin types are defined in the ASN.1 standard.

Defined types are defined by users.

The following section describes the **builtin types**. Note that the ASN.1 Object Identifier is one of the builtin types.

C.2.6.1 Any

The **Any** data type models a variable whose type is unspecified, or specified elsewhere; it can be a placeholder.

Example—
Message ::= SEQUENCE{messageId
INTEGER, messageContents ANY}

C.2.6.2 Bit String

The **Bit String** data type models binary data whose format and length are unspecified, or specified elsewhere, and whose length is not necessarily a multiple of eight.

Example—
MyFlags ::= BIT STRING
initialFlags MyFlags ::= '000'B

C.2.6.3 Boolean

The **Boolean** data type models the values of a two-state variable, for example, the answer to a yes-or-no question.

Example—
USCitizen ::= BOOLEAN
citizen USCitizen ::= TRUE

C.2.6.4 Character String

The **Character String** data type includes eight ASN.1 types: NumericString, PrintableString, IA5String, TeletexString, VideoString, VisibleString, GraphicString, GeneralString.

Example—
Emplname ::= PrintableString
sampleName Emplname ::=
“Public, John Q.”

C.2.6.5 Choice

The **Choice** data type models a variable that is selected from a collection of variables.

Example—
Myanswer ::= CHOICE{status
BOOLEAN, average REAL,
errorCode INTEGER

C.2.6.6 Enumerated

The **Enumerated** data type models the values of a variable with three or more states.

Example—
MaritalStatus ::=
ENUMERATED{single(0),
married(1),divorced(2),widowed(3) }

C.2.6.7 Integer.

The **Integer** data type models the values of a cardinal or integer variable.

Example—
StateNbr ::= INTEGER
alabama StateNbr ::= 1

C.2.6.8 Null

The **Null** data type indicates the effective absence of an element of a sequence and can be a placeholder.

C.2.6.9 Object Identifier

The **Object Identifier** data type is an identification number for data objects, the meaning of which is provided by the ISO-CCITT global naming system.

Example—
charContent OBJECT
IDENTIFIER ::= {2 8 2 6 1}
(This value identifies data objects belonging to the processable character content architecture class within the scope of ISO and CCITT standards for document architecture)

C.2.6.10 Octet String

The **Octet String** data type models binary data whose format and length are unspecified , or specified elsewhere, and whose length in bits is a multiple of eight.

Example—
ImageData ::= OCTET STRING
resetImage ImageData ::= '00'H

C.2.6.11 Real

The **Real** data type models an approximate number by expressing mantissa, base, and exponent.

Example—
SqRoot ::= REAL
sqroot2 SqRoot ::=
{141421356, 10, -8}

C.2.6.12 Selection

The **Selection** data type models a variable whose type is taken from a previously defined CHOICE.

Example—
MyAnswer ::= CHOICE{status
BOOLEAN, average REAL,
errorCode INTEGER}
MyError ::=
errorCode < MyAnswer

C.2.6.13 Sequence

The **Sequence** data type models a collection of variables whose types may be the same or different and whose order is significant. This type can be used for record definitions.

Example—
InvoiceRec ::=
SEQUENCE{vendorNbr
INTEGER, invoiceNbr INTEGER,
invoiceAmt INTEGER, etc.}

C.2.6.14 Sequence of

The **Sequence of** data type models a collection of variables whose types are the same and whose order is significant.

Example—
TiledImage ::= SEQUENCE
OF ImageData

C.2.6.15 Set

The **Set** data type models a collection of variables whose types may be the same or different and whose order is not significant.

Example—
NextEmpl ::= SET{nextEmplNbr
INTEGER, nextEmplName
PrintableString}

C.2.6.16 Set of

The **Set of** data type models a collection of variables whose types are the same and whose order is not significant.

Example—
EmplNbr ::=INTEGER
MyDept ::= SET OF EmplNbr

C.2.6.17 Tagged

The **Tagged** data type makes use of ASN.1 tags. All ASN.1 data types have numeric tags associated with them, and they are encoded into the transfer syntax. A tag consists of a *tag class* and a *tag number*. These are the tag classes:

1. Universal— These tags are defined in the ASN.1 standard. For example, the tag for data type BOOLEAN is of class Universal with a numeric value of 1, and the tag for SET is of class Universal with a value of 17. Approximately 27 Universal tags have been defined and they are globally unique.
2. Application— These tags are defined within specific ASN.1 syntaxes written for specific protocol standards and are unique within their respective standards.
3. Private— These tags are defined by specific organizations and are unique within their respective organizations.
4. Context-specific— These tags are unrestricted and are typical ly used in specific ASN.1 coding contexts to prevent ambiguities.

A tagged data type creates a new type by prefixing a new tag to an existing data type. It can be a synonym for the existing type. However, if the keyword IMPLICIT is specified, it overrides the meaning of the existing type with new semantics.

Example—
IntAppl ::=
[APPLICATION 5] INTEGER
(The INTEGER data type always has a Universal tag with the value 2. However, this production establishes an “alias” tag of class APPLICATION with the value 5).

C.2.6.18 Useful

The **Useful** data type includes *GeneralizedTime* and *UTCTime (universal time)*. These are tagged types based on the VisibleString type with special semantics. The strings must contain standardized representations of time or date information.

Example—
GeneralizedTime ::=
[UNIVERSAL 24] IMPLICIT
VisibleString
(One possible value for VisibleString is YYYYMMDD + local time.)
Example—
UTCTime ::=
[UNIVERSAL 23] IMPLICIT
VisibleString
(One possible value for VisibleString is YYMMDDD.)

A special defined type of Universal class called *External* permits the inclusion of any data value, not necessarily belonging to a valid ASN.1 data type.

C.2.7 Modules

A specific application of ASN.1 can be encapsulated as a *module*. Here is the general structure of a module.

```
<module-name> <object-identifier>
DEFINITIONS ::=
BEGIN
EXPORT
<export-objects-list>
IMPORT
<import-objects-list>
<module-body>
END
```

The export and import functions permit modules to share ASN.1 productions. For example, an ISO standard might create a new application of ASN.1 and assign it a unique module name. The new module might import existing ASN.1 productions and export its own.

C.2.8 Macros

The ASN.1 *macro* facility allows you to create new grammars for type notation and value notation. Here is the general structure of a macro.

```
<macro-name>
MACRO ::=
BEGIN
TYPE NOTATION ::=
<type-syntax>
VALUE NOTATION ::=
<value-syntax>
<supporting-syntax>
END
```

Appendix D. Overview of ISO 6523

ISO 6523 supports global naming by providing a means to identify organizations:

"In the development of this International Standard, it was recognized that a single method for identifying all organizations on an international basis was neither feasible nor practicable. Instead, this International Standard recognizes existing methods of identification and provides a means for systematically incorporating these in a uniform structure for the purpose of data interchange. In this International Standard, an organization may be identified by more than one coding method."

This standard has two main objectives— (1) specify a structure for organizations identifiers, and (2) specify administrative procedures for such identifiers. This standard introduces several key terms, including the following:

Structure for identification of organizations (SIO) Structure established by this International Standard for a unique identification of one organization among all other organizations.

International code designator (ICD) Portion of the SIO that indicates the particular organization coding system used for the purpose of unique identification of an organization when data relevant to the organization is interchanged. The ICD has no significance other than unique identification of a specific coding system.

Organization code Code assigned by an issuing organization to an organization, and unique within the particular coding system.

Organization name Name used within the SIO to verify the identity of the organization.

Issuing organization Body that assumes responsibility for the administration of a coding system for a group of organizations.

Sponsoring authority Body recognized by the requirements of this International Standard to receive proposals from issuing organizations for registration of coding systems and to submit applications accordingly to the registration authority.

The Structure for the Identification of Organizations (SIO) consists of three components:

- International Code Designator (ICD)
- Organization code
- Organization name.

The International Code Designator is a four-digit code that identifies the authority issuing the organization code. The organization code consists of 1-14 characters that uniquely identify an organization within an organization coding scheme. The organization name is a text string up to 250 characters in length. The ICD, organization code and organization name are separated by forward slash, "/." The SIO is terminated by double forward slash, "//."

ISO 6523 establishes certain functional responsibilities for maintaining this organizational naming system:

- ISO appoints a registration authority to oversee all registration activity pursuant to this standard.
- The registration authority assigns ICD codes to organizations wishing to become issuing organizations.
- The registration maintains the register of ICD codes, which contains information such as the ICD code itself, contact information for the organization granted the ICD code, the name of the coding system the organization administers, and the structure of the codes used in the organization′s coding system.
- Application for an ICD code must be supported by a sponsoring authority, which can be an ISO technical committee or subcommittee, a member body, or an international organization with liaison to ISO.
- Sponsoring authorities accept applications for ICD codes, review them, and pass approved applications to the ISO registration authority.
- Once granted an ICD code, an issuing organization undertakes the responsibility to assign and administer subordinate codes within its own coding system.

IBM is an example of an organization that has received an ICD from the ISO registration authority and has become an issuing organization with the right to issue subordinate codes. Code 0018 is the ICD code that has been assigned to IBM.

The following example illustrates the ICD process:

- Assume that the U.S. Federal Reserve Bank maintains an existing coding system for identifying U.S. banks.
- If the Federal Reserve wished its code numbers to be recognized globally, it could apply for an ICD code through the applicable sponsoring authority (ANSI).
- ANSI approves the application and forwards it to the ISO registration authority.
- ISO assigns an ICD code to the Federal Reserve.
- Participating U.S. banks, such as Chase Manhattan Bank, can be identified globally with an SIO consisting of:
  1. ICD for Federal Reserve
  2. Code issued by Federal Reserve to Chase Manhattan
  3. Organization name for Chase Manhattan.

The registration authority for ISO 6523 is the British Standards Institution (BSI).

Appendix E. Overview of X.400

X.400 is the common name for the X.400 series of Recommendations published by CCITT in 1984 and again in 1988. The 1984 version, sometimes referred to as the “red book,”contains the following Recommendations:

X.400 Describes the system model and service elements

X.401 Describes the basic service elements and optional user facilities

X.408 Describes rules for data type conversions

X.409 Describes a special notation for defining data types

X.410 Describes remote operations and reliable transfer service elements

X.411 Describes the P1 and P3 protocols, service primitives and protocol data units (PDUs)

X.420 Describes the P2 protocol, service primitives and PDUs

X.430 Describes access protocol for Teletex terminals.

The 1988 version, sometimes referred to as the “blue book,” contains the following Recommendations:

X.400 Describes the system model and service elements

X.402 Describes the overall architecture

X.403 Describes methodology for conformance testing

X.407 Describes abstract service definition conventions and notation

X.408 Describes rules for data type conversions

X.411 Describes the Message Transfer System in terms of an abstract service definition

X.413 Describes the Message Store in terms of an abstract service definition

X.419 Describes the P1, P3, and P7 protocols in terms of protocol specifications

X.420 Describes the Interpersonal Messaging System (IPMS), including the P2 protocol.

The overall purpose of the X.400 standard is to describe a *Message Handling System (MHS)* capable of transferring electronic messages from originators to recipients through a store-and-forward network. The *Message Transfer System (MTS)* consists of functional objects called *Message Transfer Agents (MTAs)* and is the portion of the MHS that accepts messages from originators, transfers messages among MTAs, and finally delivers messages to recipients. Originators and recipients are served by functional objects called *User Agents (UAs)*, which can support end users but are not identical with end users. When an originator causes an electronic message to enter the MTS, this is called message submission. When an MTA causes an electronic message to reach a recipient, thus leaving the MTS, this is called message delivery.

Protocols were created to support various kinds of communication between functional objects within the MHS.

- The P1 protocol specifies how MTAs communicate with each other.

- The P2 protocol specifies how UAs communicate with each other using IPMS.
- The P3 protocol specifies how MTAs communicate with UAs or Message Stores.
- The P7 protocol specifies how UAs and Message Stores communicate with each other.

The *Message Store* was added in the 1988 version of X.400. It is a functional object logically located between an MTA and a UA and acts as a repository for delivered messages, keeping them available until the UA chooses to deliver them.

Index

A

Abandon directory operation
  See X.500 Directory, Operations
absolute time 127
Abstract Syntax Notation One (ASN.1) 78
  overview 135, 141
Add Entry directory operation
  See X.500 Directory, Operations
Advanced Peer-to-Peer Networking
  See APPN
Alias names
  See X.500 Directory
ANR 47
  label 48
APPN 42
  end node 42, 49
  network node 43, 49
APPN node 49
APPN or LEN node 49
APPN VTAM 51
ASN.1
  See Abstract Syntax Notation One (ASN.1)
Atomic Time 127
authorized end node 50
automatic network routing
  See ANR

B

border node 53
  extended border node 53
  peripheral border node 53
boundary node 51

C

CCITT X.500/ISO 9594 58
CDS
  See Cell Directory Service (CDS)
CDS_LinkTarget 69
CDS_Replicas 70, 71
Cell Directory Service (CDS) 55
central resource registration 49
chaining
  See X.500 Directory
Chaining Prohibited
  See X.500 Directory, Service Controls
child pointer 65, 66, 69, 70, 71
class of service 49
clearinghouse 62, 64, 65, 66, 67, 68, 69, 70, 71
clock inaccuracy
  See Open Software Foundation Distributed Computing Environment (OSF DCE), clock inaccuracy
CNN 44, 51
Compare directory operation
  See X.500 Directory, Operations
composite network node 44, 52
  See also CNN
composite node 51
connection network 53
Courier Time Server 96
CP-CP session 52
CP-CP sessions 53
cross reference
  See X.500 Directory

D

DAP
  See Directory Access Protocol (DAP)
DCE
  See Open Software Foundation Distributed Computing Environment (OSF DCE)
DCE Distributed Time Services (DTS)
  See Open Software Foundation Distributed Computing Environment (OSF DCE), DCE Distributed Time Services (DTS)
DECdns 56
DIB
  See Directory Information Base (DIB)
Directory
  white pages 3
  yellow pages 3
Directory Access Protocol (DAP) 17
Directory Information Base (DIB) 16, 17
Directory Information Tree (DIT) 16, 18
Directory Service Agent (DSA) 16
Directory Service Protocol (DSP) 16
Directory User Agent (DUA) 17
Distinguished Name (DN) 21
Distributed Computing Environment (DCE)
  See Open Software Foundation Distributed Computing Environment (OSF DCE)
DIT
  See Directory Information Tree (DIT)
DN
  See Distinguished Name (DN)
DNS
  See Internet, Domain Name System (DNS)
domain 48
Domain Name System (DNS) 58
  See also Internet, Domain Name System (DNS)
DSA
  See Directory Service Agent (DSA)
DSP
  See Directory Service Protocol (DSP)
DTS Daemon 102, 112

dtscp 102, 103, 104
dtsd 102, 103
DUA
  See Directory User Agent (DUA)

E

end node 49
enhanced session address 48
epoch 104
extended border node 51, 53

G

GDA
  See Global Directory Agent (GDA)
GDS
  See Global Directory Service (GDS)
Global Directory Agent (GDA) 58, 59
Global Directory Service (GDS) 55, 58, 61
Global Time Server 96, 101
GOSIP
  See Government OSI Profiles (GOSIP)
Government OSI Profiles (GOSIP) 40
Greenwich Mean Time (GMT) 105

H

HPR 47
  addressing 48
HPR node 51

I

ICD
  See International Code Designator (ICD)
inaccuracy (clock)
  See Open Software Foundation Distributed Computing Environment (OSF DCE), clock inaccuracy
interchange node 51, 52
intermediate node 42
intermediate session routing 49
International Code Designator (ICD) 83
International Time Bureau (BIH) 105
Internet
  ARPANET 10
  Domain Name System (DNS) 9, 12
  HOSTS.TXT 11
  TCP/IP 10
ISO 3166 83
ISO 639 83
ISO 6523 83
ISO 8824 78
ISO 8879 82
ISO 9070 82
ISO 9594
  See X.500 Directory
ISO 9834 80

L

leap seconds 100, 127
leap year 127
LEN 41
  connection 41
  end node 41, 50
List directory operation
  See X.500 Directory, Operations
Local Scope
  See X.500 Directory, Service Controls
Local Time Server 96
logical unit
  See LU
lookup 64, 65, 70
low-entry networking
  See LEN
LU 41

M

middleware 6
Modify Entry directory operation
  See X.500 Directory, Operations
Modify RDN Entry directory operation
  See X.500 Directory, Operations
multicasting
  See X.500 Directory
  See X.500 Directory, multicasting (parallel chaining)
MVS Time
  See Time

N

NADF
  See North American Directory Forum (NADF)
NAU 46
net ID 46
  registry 46
network accessible unit 46
network address 46
network identifier 46
Network Information Center (NIC) 60
network name 46
network node 49
network node server 49
network-qualified name 46
Networking Blueprint 5
NIC
  See Network Information Center (NIC)
node
  HPR node 53
North American Directory Forum (NADF) 40
Novell Netware Directory Services (NDS) 118

O

Open Blueprint 5, 54, 74, 95
Open Blueprint Directory 54
Open Blueprint Naming 74
Open Blueprint Time 95
Open Software Foundation Distributed Computing Environment (OSF DCE)
  clock inaccuracy 99, 111
  DCE Directory 10
  DCE Distributed Time Services (DTS) 96
  Directory 56
  Remote Procedure Call (RPC) 55
OSF
  See Open Software Foundation Distributed Computing Environment (OSF DCE)

P

parallel chaining
  See X.500 Directory, multicasting (parallel chaining)
peripheral border node 51, 53
peripheral node 51
Prefer Chaining
  See X.500 Directory, Service Controls

R

RDN
  See Relative Distinguished Name (RDN)
Read directory operation
  See X.500 Directory, Operations
referral mode
  See X.500 Directory
Relative Distinguished Name (RDN) 21, 22
reltimespec 108, 109
Remote Procedure Call (RPC) 55
Remove Entry directory operation
  See X.500 Directory, Operations
Replicas 64
RPC
  See Remote Procedure Call (RPC)
RTP 48

S

SATF 53
Schema
  See X.500 Directory
Search directory operation
  See X.500 Directory, Operations
Search Guide
  See X.500 Directory
ServerRequestProviderTime 112
session
  identifier 47
SGML
  See Standardized Generalized Markup Language (SGML)
Siemens 56
Size Limit for List and Search
  See X.500 Directory, Service Controls
SNA 42
soft link 65, 68, 69
SSCP 48, 51
SSCP-SSCP session 52
Standardized Generalized Markup Language (SGML) 82
subnet
  HPR subnet 54
subnetwork 53
subordinate reference
  See X.500 Directory
superior reference
  See X.500 Directory
Sysplex Timer 127
  See also Time
system network architecture
  See SNA
system services control point
  See SSCP

T

T2.1 node 49
TCP/IP
  See Internet
Time
  in MVS 130
  MVS Time 96
  Sysplex Timer 96
  technology 95
Time Clerk 96, 100
time in MVS 127
TIME macro 131
Time Provider Interface (TPI) 96, 111
Time Representation 96, 105, 106
Time Servers 96, 101
timespec 108, 109, 110
timestamp 131
TOD 127
topology subnetwork 53
TP Process 112
transmission header 47

U

unauthorized end node 50
utc 108, 109, 110, 111, 127
utc_mkreltime 110

V

virtual routing node 51, 53

W

white pages 3

X

X.400 145
  overview 145, 146
X.500 Directory 58, 59, 60
  Alias names 19
  chaining 32
  cross reference 36
  Directory Access Protocol (DAP) 17
  Directory Information Base (DIB) 17
  Directory Information Tree (DIT) 18
  Directory Service Agent (DSA) 16
  Directory Service Protocol (DSP) 16
  Directory User Agent (DUA) 17
  Distinguished Name (DN) 21
  Government OSI Profiles (GOSIP) 40
  multicasting 17
  multicasting (parallel chaining) 32
  North American Directory Forum (NADF) 40
  Operations
    Abandon 27
    Add Entry 30
    Compare 28
    List 28
    Modify Entry 31
    Modify RDN 31
    Read 27
    Remove Entry 30
    Search 29
  referrals 17, 33
  Relative Distinguished Name (RDN) 21, 22
  Schema 22
  Search Guide 37
  Security 38
  Service Controls 36—37
    Chaining Prohibited 36
    Don’t Dereference Aliases 37
    Don’t Use Copy 37
    Local Scope 36
    Prefer Chaining 36
    Priority 37
    Scope of Referral 37
    Size Limit 37
    Time Limit 37
  Subordinate Reference 35
  Superior Reference 34

Y

yellow pages 3

ITSO Technical Bulletin Evaluation RED000

The Library for System Solutions
Directory, Naming and Time

Publication No. GG24-4104-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction	____		
Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____ _____
Name Address

_____ _____
Company or Organization

_____ _____
Phone No.

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Mail Station P099
522 SOUTH ROAD
POUGHKEEPSIE NY
USA 12601-5400

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.