S. Snyder (IBM)
June 1992

DCE SIG NAMING REQUIREMENTS

1. INTRODUCTION

The DCE Naming workgroup is a sub-group of the OSF DCE Special Interest Group whose focus is on the Naming components of the OSF's DCE as well as system wide naming issues. Specifically, its CHARTER is as follows:

(a) The DCE Naming Workgroup is an advisory body which generates recommendations to the OSF on strategy and direction, requirements and architectural and design changes to the Naming components of the OSF DCE. These recommendations will include (but are not necessarily limited to):

  (i) Detailed requirements papers — providing prioritization and interpretations of sets of requirements.

  (ii) Recommendations for architectural work/papers to flesh out high level impacts of a specific set of requirements.

  (iii) Reviews and positions on design proposals to be implemented in a release of the DCE.

(b) It is explicitly NOT within the charter of the Naming workgroup to actually generate the designs as part of their working meetings (design by committee).

This document represents the first major deliverable from the Naming Workgroup to the OSF. It describes the Working Group's recommendation for the evolution of the DCE Naming Services from their 1.0 state through DCE 2.0. These recommendations are based on the feedback received on the DCE SIG Ballot as well as the technical expertise of the Working Group's membership.

This document is organized into three major sections and some appendices. This first section is introductory. The second section describes the process through which the requirements were generated and prioritized. The third section, which is the main section of the document, presents the results of this process, in terms of recommendations to the OSF on long term direction and delivery contents. The appendices contain, respectively: a summary of the naming portion of the DCE Ballot results and the naming specific comments included by the respondents; the detailed descriptions of the ballot items which were not explicitly included in the recommendations; the change history for this document; the membership of the DCE Naming workgroup when this DCE-RFC was posted.

2. APPROACH

The process of developing an overall SIG recommendation for enhancements to DCE 1.0 was driven by the DCE SIG's Requirements ballot process. Each of the DCE working groups generated a set of proposed architectural, functional and implementation enhancements to DCE 1.0. These lists were then combined into a single ballot which was voted on by the DCE SIG membership. The initial naming requirements input to this process combined with the results of the voting (the naming portion of which is summarized in Appendix A) formed the basis for the recommendations made by this paper.

Because of the number and scope of naming requirements included in the ballot, the following process was used to interpret the ballot results into a rational set of recommendations:

(a) The ballot ratings for each item were compared across both the 1.1 and 2.0 timeframes in order to identify the highest priority set of requirements. This was done across both releases (as opposed to looking at 1.1 and 2.0 separately) so that those highly rated 2.0 items which require a significant amount of development work were not serialized behind a long list of 1.1 items. These high priority items were then further organized according to steps 2 and 3 below. The rest of the requirements were considered deferred and will not be addressed further in this document (the detailed description of the deferred requirements are included in Appendix B for completeness).

(b) The high priority set of requirements was further divided into short term and long term categories based on:

(i) 1.1 priority from ballot results
(ii) Dependencies on other changes/enhancements
(iii) Technical complexity
(iv) etc.

This step identified those high priority items which could reasonably be expected to make the DCE 1.1 release, which could be staged across 1.1 and 2.0 releases and which items should be targeted for DCE 2.0.

Given the time frame of 1.1 (which is currently targeted for DCE 1.0 + 18 months), and the fact that (at the writing of this paper) the DCE 1.0.1 content was ''closed'', the work group also identified a set of requirements as *pre-1.1*. These requirements were viewed as being necessary changes prior to the 1.1 timeframe, but acknowledged as not feasible for inclusion in 1.0.1. The working group did not provide any specific recommendation as to a release vehicle for these changes.

(c) Those items which were placed in the 2.0 timeframe were examined to identify those whose implementation were viewed as technically difficult enough to warrant the recommendation that resources be applied to them in the 1.1 timeframe.

The output of this process is detailed in the Recommendations section of this document, which follows.

3. RECOMMENDATIONS

This section contains the specific recommendations of the Naming workgroup to the OSF for the evolution of the DCE Naming Services. The original objective of the process through which it was generated was to provide input on the content of the DCE 1.1 and 2.0 releases. However, in the course of generating that set of recommendations, additional recommendations were developed addressing both long term directional and short term pre-1.1 issues. All of these recommendations are presented in the sections which follow.

3.1. Strategic Recommendations

This section contains a set of strategic statements regarding the longer term direction of the OSF DCE Naming services which the working group recommends the OSF adopt as the basis for direction of future work on the DCEs Directory Services.

3.1.1. Architectural objectives

This section defines the set of architectural objectives which will drive the evolution of the DCE Naming Services. The objectives are divided into three categories to address the Naming Model, Name Services and Composite Name Space support elements of the DCE Naming Services architecture. For background, see [Mart 91], [OSF 90a], [OSF 90b], [OSF 90c].

3.1.1.1. Naming model

The Naming Model element of the DCE Naming Services architecture defines the key characteristics of ''names'' within the DCE.

(a) *Single name space*

All users of an object use the same name for an object, regardless of the location of the object or the user of the name. This allows names to be communicated between users and applications outside of the context of the DCE Naming Services and still retain their meaning which simplifies the process of sharing information in a distributed environment.

(b) *Universal name space*

The single name space must be capable of encompassing all types of objects, including files, services, user, groups, processes, etc. This allows objects to be named in a uniform way and helps enable the development of generic tools which can operate on a variety of objects.

(c) *Global name space*

The single name space must be scalable up to allow the assignment of names across a world wide network of systems and down to accommodate a small LAN network environment. Additionally the name space must support the delegation of naming authority (roughly equivalent to the ability to create/assign names to objects) so that there is no single central entity which must have knowledge of all of the names in the system.

3.1.1.2. Name services

The Name Services element of the DCE Naming Services architecture defines the *scope* of naming functionality supported by the DCE Naming Services and a set of services which provide ‘‘generic’’[1] support for that functionality. The *scope* of naming functionality refers to the maximum functionality supported at the DCE Naming APIs and basically defines what functionality is considered to be ‘‘naming’’ functionality and what is not. This concept becomes important in the discussion of Composite Name Space support below.

(a) *Scope of functionality*

The scope of functionality supported by the DCE Naming Services is that defined by the X/OPEN Directory Services API.[2]

(b) *X/OPEN XDS API support*

The X/OPEN Directory Service API will provide access to all of the DCE Naming Services functionality. This does not imply that all elements of the DCE Naming Services will provide support for the full scope of functionality defined by the XDS interface. It does mean that when naming function is provided, it accessible through the XDS interface.

(c) *Security*

The DCE Name Services must provide secure access information they contain in a manner which is integrated with the DCE Security Services.

(d) *Reliability*

Because of its central position in the system, the DCE Name Services must be highly available despite the presence of machine failures and network partitions. This availability is achieved through replication: copies of entries are maintained by multiple servers that are distributed across the network so that single points of failure for naming operations are avoided.

(e) *Performance*

To provide acceptable performance, the name services must implement caching, so that most requests do not have to go out over the wire.

(f) *Access to X.500*

To achieve truly global interoperability among multiple cells and other environments, the DCE Name Services must provide access to X.500 directory protocols.

3.1.1.3. Composite name space support

While the Naming Model objectives target a single, universal name space which encompasses all types of objects, the implementation characteristics of most ‘‘generic’’ name services (weakly consistent replication, the assumption of a slowly changing name space) do not make any one of them suitable for naming all objects likely to part of a distributed computing environment. Therefore, the OSF DCE Naming Architecture must provide a well defined mechanism through which objects with different naming characteristics can be seamlessly incorporated into the global namespace.

---

1. ‘‘Generic’’ in the sense that these services do not provide functionality in addition to their naming functions against the names they export (e.g., stream I/O, etc.).

2. Note that ‘‘functionality’’ refers to semantics, and does not imply syntax.

Seamlessness, in this context, is defined as follows:

(a) All object names exported via the mechanisms described above must be ‘‘resolvable’’ via a single name resolution function. Resolvable refers to the ability to derive the location and potentially some other minimal set of information about the object from the objects name.

(b) All naming functionality (as defined by the Scope of Functionality in the previous section) supported by objects whose names are exported through this mechanism must be accessible via the XDS application level programming interface. Note that this does not imply that all objects exported into the global namespace must necessarily support all of the function defined by the XDS API, but rather that the subset of function that is supported by exported through the XDS API.

This capability, referred to here as *Composite Name Space Support:* will allow resource managers (a.k.a. object managers, servers, etc.) to participate directly and in a consistent fashion in the global namespace. Furthermore, the provision of a single naming programming interface at the application level which can be used to perform basic naming functions across all names, enables broad set of application function including the development of generic namespace browsers.

3.1.2. X.500/CDS Relationship

This section addresses the relationship between the GDS (X.500) and the CDS core name services.

The X.500 compliant Global Directory Service (GDS) and the Cell Directory Services (CDS) address overlapping but different architectural objectives within the DCE Naming Architecture. The principal focus of the GDS is to provide fully compliant access to X.500 Directory Services to applications which are running in an OSF DCE environment. Additionally, the GDS provides the means through which DCE Cells can be integrated into a global name space rooted in an X.500 Directory Service. The CDS on the other hand is focussed on providing a light-weight implementation of directory functionality which is tightly integrated with the rest of the DCE environment.

Given that the X.500 Directory Services standard and the OSF DCE are continuing to evolving independently of each other, the need for two ‘‘generic’’ directory implementations which maintain their respective focuses will continue to exist in order to successfully satisfy all of the requirements for naming for users of the DCE. It is also important, however, that for the functionality which is provided by both services, seamless access to that functionality be supported through the XDS API. Furthermore, additional function which can be supported by the CDS without sacrificing its local cell naming objectives should be added to further improve the functional transparency provided by the XDS API.

Although CDS is intended for use in the cell and is tightly integrated with other DCE components, we believe that there a need for accessing X.500 services by applications running in a cell. For example, a DCE mail application may wish to access the names of mail recipients from the Directory, e.g., to read an X.500 name in order to send mail to an O/R name contained as an attribute. Also mail applications running in non-DCE environments should be able to access names of DCE users by looking them up in the global Directory. These capabilities have been referred to colloquially as ‘‘bringing X.500 into the cell’’.

Unfortunately, X.500 is not currently nor will it ever be integrated with other DCE services to the level of CDS. However, a naming requirement contained elsewhere in this document describes the need to allow access to X.500 using a DCE principal identity. Other integration requirements should also be considered, e.g., to allow the mapping of a login/principal name to an X.500 name and vice versa.

3.2. Delivery Recommendations

The following table summarizes the working group’s delivery recommendations:

Requirement	Pre 1.1	DCE 1.1	DCE 2.0
Decouple XDS from GDS	X		
Blocking reentrant XDS/XOM libraries	X		
Code cleanup	X		
Seamless XDS API to CDS and X.500		X	
Minimal XDS search capability in CDS		X	
Exploit threads in X.500 client libraries		X	
CDS administrative improvements		X	
Serviceability		X	
Performance enhancements		X	
Internationalization		X	
Support for hierarchical cells		X	
Support for string names in XDS/XOM		X	
Access to X.500 using DCE identity		X	
Specifications for major DCE 2.0 requirements		X	
DME integration for management operations			X
Support for 1992 X.500 standard			X
Single naming interface			X

Table 1. Delivery Recommendations Summary

The requirements described in this document and identified in Table 1 above are derived from but are not identical to the Naming Requirements which appeared in the DCE SIG Ballot. In the process of placing them into a cohesive set of timetables, some of the original requirements have been broken into multiple requirements in order to facilitate staging the requested function across multiple releases and some of the requirement descriptions have changed as the work groups understanding of them has grown through the discussions associated with the development of this paper. However, meeting the ultimate goal of producing a reasonable approach to satisfying the most important requirements as identified by the DCE SIG Ballot has (hopefully) only been helped by these refinements.

A number of the requirements shown above are not Naming specific requirements, but rather are high priority general DCE requirements which will have impacts on the Naming components. The general objectives of these requirements are being developed by other working groups (e.g., the I18N working group), and detailed changes required to the naming components will be developed by the component technology providers based on those objectives.

The remainder of this section contains the detailed descriptions of the requirements listed in the table above.

3.2.1. Pre-1.1 requirement descriptions

The requirements described in this section are considered to be necessary in a timeframe prior to the currently targeted DCE 1.0 + 18 months for DCE 1.1. Since the content of DCE 1.0.1 is already closed, and there is currently no official plan for a 1.0.2, there is no specific delivery vehicle for these requirements. However, by identifying these ahead of the DCE 1.1 timeframe, the working group is asking the OSF to consider making these functions available (through some means) prior to 1.1.

3.2.1.1. REQT: Decouple XDS from GDS

Currently, an application is not able to contact CDS over XDS when GDS is not running and therefore, the XDS programming interface is only available when the GDS client processes are installed on the machine. Since GDS is not a core service (and therefore may not be installed on any given machine), there will not necessarily be a standard interface available for accessing the CDS.

This requirement would correct this situation by decoupling the XDS interface from the GDS such that the XDS interface could be installed independently from the underlying directory services. In particular, the XDS library should support the following configurations on any given machine:

(a) XDS and the Core Services (CDS, RPC, TIME and SECURITY)
(b) XDS and GDS (No Core Services present)
(c) XDS and both the Core Services and GDS

3.2.1.2. REQT: Blocking reentrant XDS/XOM libraries

The DCE 1.0 version of the XDS/XOM libraries are not threadsafe. As a result, they do not allow multithreaded applications to invoke the interface from multiple threads. Furthermore, since the XDS library does not support the optional asynchronous invocation features of the X/OPEN XDS specification, there is no straightforward mechanism for asynchronous invocation of XDS operations.

This requirement would correct this situation by making modifications to the XDS and XOM libraries to make them (at a minimum) blocking reentrant. Note that this requirement is not asking for complete support of the threading model throughout the GDS client stack, but only for enough support to allow multithreaded applications to be written to use the interface and not be explicitly aware of its nonrentrant implementation. (Note: GDS exploitation of the DCE threads is addressed in the ‘‘Exploit threads in X.500 client libraries’’ requirement.)

3.2.1.3. REQT: Code cleanup

The following code cleanup requirements should be included in the DCE code base as soon as possible in addition to the DCE-wide code cleanup items identified in the DCE ballot.

(a) GDS specific
    (i) Remove use of macros (such as `If`, `Fi`) that obfuscate the code.

(b) CDS specific
    (i) Remove the use of the all_ptr union from the CDS code base.

3.2.2. DCE 1.1 requirement descriptions

The following items are recommended for inclusion in DCE Release 1.1.

3.2.2.1. REQT: Seamless XDS API to CDS and X.500

Programmers use XDS for accessing the DCE Directory Services as a single service. The fact that there are different directory service implementations (X.500 and CDS) behind the XDS interface should be fully transparent to the programmers for functions which are supported by both services. In DCE 1.0, this is not the case. The following is a list of functions/capabilities that are not available to programmers when target operation is to be performed by CDS rather than GDS.

(a) *Tree construction*

The CDS does not support the creation of ‘‘directories’’ (nodes in the name space which can have subordinates) via the XDS interface. For example, a program attempting to perform the following two legitimate XDS operations in sequence would fail:

```
ds_add_entry /.../C=US/O=xyz/a
ds_add_entry /.../C=US/O=xyz/a/b
```

(b) *Rename (Modify_RDN)*

(c) *Atomic operations across multiple attributes*

All of the discrepancies are visible to and must be dealt with by the application programmer using XDS. More significantly, these differences unnecessarily limit the portability of XDS compliant applications onto the DCE XDS API because functions which are supported by the CDS are in fact not exported through the XDS API.

3.2.2.2. REQT: Minimal XDS search capability for CDS

This requirement proposes a minimal subset of the XDS Search capability to be supported by the CDS. It does not suggest that the search capability of CDS be limited to this level forever, but rather recommends an initial set of basic function which would be beneficial to programmers without requiring the implementation complexity of the full XDS search semantics. Future releases of DCE may support additional search capability if the requirement is there.

The proposal looks at what could be a minimal subset and still be usable by those applications that need a simple search capability to retrieve a small set of name service entries that meets certain simple criteria. The intended use of this subset search capability is primarily for clients to look up appropriate servers based on certain attribute-values.

(a) *Level of Search*

XDS Search, as specified in the X/OPEN CAE, allows three levels of search. These are:

(i) Base object — Search just the given object entry
(ii) One level — Search just the immediate subordinates
(iii) Whole subtree — Search the whole subtree including the subtree root.

The proposed minimal subset is to allow a single-level search only. This would limit the search to one replica (directory) in CDS and certainly limit it to one clearinghouse and server.

This is specified by the ‘‘subset’’ argument of the XDS Search function call.

(b) *Search_aliases*

This argument specifies whether any aliases in the subordinate entries being searched should be dereferenced.

The proposed minimal subset is to support no dereferencing of alias in the subordinate entries to keep the search within the replica.

(c) *Filter*

The filter is used to eliminate entries from the search that are not wanted. Information will only be returned on entries that satisfy the filter.

This argument is an instance of OM_Class Filter. An instance of this OM_Class has the following OM_attributes:

(i) Filter-Items — 0 or more values
(ii) Filters — 0 or more values
(iii) Filter-Type — AND, OR, NOT

The proposed minimal subset is to support 0 to <n> number of Filter-Items, zero Filters (i.e. no nested component filter) and Filter-Type of AND only. (Where <n> is a small integer.)

(d) *Filter-Item-Type*

Filter-Item-Type is part of the Filter-Item attribute of the Filter argument above. The X/OPEN CAE specifies the following Filter-Item-Type:

(i) approximate-match
(ii) equality
(iii) greater-or-equal
(iv) less-or-equal
(v) present
(vi) substrings

The proposed minimal subset is to support Filter-Item-Types of ‘‘equality’’, ‘‘present’’ and ‘‘substrings’’ only.

The ‘‘substrings’’ filter-item-type would be useful for matching attribute values (of a specified attribute type) that are made up of several related sub-fields.

(e) *Other Arguments*

All other arguments for Search are handled in full as per the X/OPEN CAE specification (except the synchronous/asynchronous Service_Control which DCE currently only supports the synchronous mode of operation for all XDS calls).

This proposed subset will allow an application to search for all entries in a CDS directory that are of a certain Object Class and with attribute XXX present (for example).

Any search request outside of the allowable subset directed to CDS will result in an error (Unwilling-to-perform) as in DCE 1.0 now.

3.2.2.3. REQT: Exploit threads in X.500 client libraries

This requirement would extend the GDS client to fully exploit the threaded DCE environment.

3.2.2.4. REQT: CDS administrative improvements

These improvements would be geared towards simplifying the administrative tasks associated with maintaining an operational CDS. Such enhancements would include:

(a) *Allow replication sites for new directories to be inherited from the parent directory.*

Whereas in DCE 1.0 the replication site for each newly created directory must be explicitly specified, this enhancement would allow the administrator to explicitly specify the replication sites for a subtree of directories only once (for the root directory of the subtree) and all subsequently created directories in the subtree would automatically be replicated to the same sites.

(b) *Support for bulk administrative operations:*

(i) Rename/move a subtree
(ii) Delete a subtree
(iii) Copy a subtree
(iv) Replicate a subtree

Initial support for these operations should at a minimum guarantee that they are performed atomically (i.e., they either complete successfully or no changes are made at all). Full support includes the capability to perform these operations without taking any of the replicas offline.

3.2.2.5. REQT: Serviceability

This requirement comes from the high ballot rating for the DCE in general to provide better serviceability support. The naming components will support the general DCE serviceability enhancements are defined for 1.1. Recommendations for those objectives are being driven through the DCE Serviceability working group.

3.2.2.6. REQT: Performance enhancements

This requirement reflects the high ballot rating for an overall DCE performance improvement requirement. Requirements for specific changes to the naming components are pending the development of baseline performance numbers and performance objectives for DCE in general and for the naming components specifically. Work on developing these objectives is being driven out of the SIG DCE Performance working group.

3.2.2.7. REQT: Internationalization

This reflects the high ballot rating for the DCE in general to provide better internationalization support. Detailed requirements specific to naming for this item are dependent on the DCE objectives for Internationalization in the 1.1 timeframe. Recommendations for those objectives are being driven through the SIG DCE I18N working group.

3.2.2.8. REQT: Support for hierarchical cells

Hierarchical cells are cells whose root directory is cataloged in another CDS server. Nesting can be quite deep, but ultimately, the tree should be rooted (cataloged) in the Global Namespace. The advantage of this is that cell administrators do not have to allocate names in the Global Namespace and hierarchical cells for the basis for hierarchical trust agreements. For example, assume that DEC is composed of a set of hierarchical cells. The name of the uppermost cell in the hierarchy is `/.../dec.com`. The `lkg` cell, named `/.../dec.com/lkg`, is rooted in `/.../dec.com`, as is the `zko` cell `/.../dec.com/zko`. We assume that the administrators of the `lkg` and `zko` cells have exchanged keys with the administrator of `dec.com`. Because of hierarchical trust, the administrators of `lkg` and `zko` do not have to exchange keys among themselves to allow `lkg` users to read `zko` files and *vice versa*. Since they both trust a common ancestor, they automatically trust one another.

3.2.2.9. REQT: Support for string names at the XDS/XOM API

Currently the XDS API supports the input of entry names as objects of OM class Name. This is an abstract class, i.e., there are no specific representations of Name defined in the spec. Currently there exists only one concrete subclass, that of DS-DN.

DS-DNs are composed of 0 or more OM Attributes called RDN. RDN's are themselves OM objects and are composed of one or more OM attributes called AVA's. AVA's are also objects and are composed of two OM attributes: Attribute-Type, an OM String representing an object identifier, and Attribute-Value, which can have any value. As should be evident, supporting the full generality of X.500 names requires a complicated structure, even by typical XDS/XOM standards.

A simpler representation is required to increase the usability of XDS. The API specification states: ''It is expected that vendors will define additional subclasses to provide alternate representations of names.'' We propose that the OSF standardize a new subclass of Name. Its representation should consist of an OM String. This string would be of the form of all other DCE string names as documented in the ''Introduction to OSF DCE'' and in the same representation as string names passed to all other DCE APIs.

3.2.2.10. REQT: Access to X.500 using DCE identity

Currently in DCE, when a user tries to access a resource with a global name, they may be asked to enter an X.500 userid and password information before his local DUA will be able to proceed with trying to resolve the name. This characteristic is a result of the different authentication mechanisms used by the cell-based DCE services and the X.500 Directory Service. This requirement asks for modifications to the DCE X.500 server implementation such that DCE X.500 servers can be configured to participate in the DCE authentication mechanism. This change would simplify the administrative work involved with using X.500 as the GDS since they would no longer have to define and maintain multiple definitions and passwords for each user. Further, users would no longer have to deal with remembering and maintaining separate passwords for X.500 and DCE.

Note: While this proposal would eliminate the problem of logging onto each server within a given enterprise or installation that used all DCE X.500 servers, it will not (can not) solve the general case problem because the X.500 standard does not specify the same authentication mechanism as is used by DCE. Therefore, a non-DCE X.500 server which uses a different authentication scheme will still require administration of a separate user definition than that defined for DCE services.

3.2.2.11. REQT: Specifications for major DCE 2.0 requirements

This requirement represents the SIG DCE Naming working group's concern that work begin in the short term to evolve the DCE Naming architecture towards the objectives described in the section on ''Architectural Objectives'' (above). In particular, by placing this requirement in the 1.1 timeframe the working group is asking the OSF to apply resources in the 1.1 timeframe to the development of the 2.0 design.

3.2.3. DCE 2.0 requirement descriptions

The following are requirements recommended for inclusion in DCE 2.0.

3.2.3.1. REQT: DME integration for management operations

There are several components of the CDS implementation which provide ''management'' capabilities as they are defined by the OSF DME selection. This requirement proposes modifying these components of the CDS to integrate them with the appropriate DME services. In particular:

(a) Integrate the diagnostic and control programs with the DME application services.

(b) Modify the event logging functions of the CDS to make use of the DME event management services interface.

(c) Integrate the CDS Browser with the DME graphical user interface.

3.2.3.2. REQT: Support for 1992 X.500 standard

Need to upgrade the DCE Directory Service to support the 1992 X.500 standards when it is approved. Some of the major additions to the X.500 standards in 1992 include:

(a) Extended directory information model
(b) Extended schema
(c) Replication
(d) Access control
(e) Etc.

Implementation of the 1992 standards will allow OSF DCE Directory Service to interoperate fully with other non-DCE X.500 Directory Services that are compliant with the 1992 standards.

3.2.3.3. REQT: Single naming interface

This item represents a proposed architectural change to the current DCE Directory/Naming structure which would enable the development of generic, cross resource manager (and object class) tools and applications. One example of such a application is a generic name space browser which could display, browse (navigate), and manipulate names and attributes across all resource managers exporting names into the global namespace.

In the DCE 1.0 architecture, each resource manager which exports names and attributes into the global namespace must define its own specialized interfaces and protocols for mapping names onto locations, functionality and attributes. Additionally, each resource manager client must implement a ‘‘junction’’ between the CDS and the set of names exported by its servers. These junctions allow the resource manager clients to resolve the GDS and CDS portions of a global name using the XDS or CDS APIs, and then switch to their own protocol to resolve the rest of the name. Since these junctions are specific to each resource manager, there is no single interface through which basic naming functions are provided against all names in the global namespace. As a result, truly generic, cross resource manager applications cannot be easily supported because they must understand and use the specific interfaces for each resource manager they wish to support. Whenever a new name exporting resource manager is developed, all of the cross resource manager applications must be updated to utilize that resource manager’s specific naming interfaces.

In order to enable the development of truly generic, cross resource manager tools and applications, the following architectural modifications/additions to the current DCE Naming architecture are proposed.

(a) A single set of remote interfaces which can easily be exported by all resource managers wishing to export names into the global namespace. These interfaces would provide an open, extensible DCE Directory/Naming architecture and should have the following characteristics:

  (i) Provide a clearly specified protocol that could be exported by a variety of resource managers.

  (ii) Support a range of functional sets from simple to more complex operations.

  (iii) Include conventions for adding and removing pieces of the namespace (i.e., for splicing different pieces of the namespace together in a way that is convenient for the name exporting resource managers and that is transparent to their clients).

  (iv) Provide access to functionality which is equivalent to that defined by X.500.

  As part of the protocol definition, the architecture should define the minimum set of functionality that MUST be supported by a resource manager in order for it to be integrated into the global namespace and be accessible through the common system level programming interface. Examples of a possible minimum functional set would include a path resolve function which would allow a name to be completely resolved to the location of the named object using a standard referral mechanism and the functions required to splice name spaces together (e.g., add name, delete name, list names, etc.).

(b) A single set of system level programming interfaces which provide basic naming, lookup, and attribute functions across all names in the global name space. This set of interfaces should have the following characteristics:

  (i) Provide access to all existing DCE name exporting resource managers (i.e., X.500, CDS, DFS, Registry) which have been extended to export the naming interfaces defined in the previous point above.

  (ii) Require no modifications when new, compliant resource managers are added to the namespace.

(iii) Provide a simpler (relative to XDS) programming model for basic, frequently performed naming operations (for example, map a name onto a set of interface UUID's associated with the name).

(iv) Provide access to the functionality equivalent to that defined by X.500.

4. ACKNOWLEDGEMENTS

General thanks are due to all the members of the DCE SIG Naming Sub-group for their valuable insights, comments and time in reviewing the several versions of this document and ultimately providing a valuable set of requirements information to the OSF. Special thanks are due to Mark Fox (DEC), Liza Martin (HP), and Ray Kwok (IBM) for their significant textual contributions to the paper.

APPENDIX A. NAMING REQUIREMENTS — BALLOT RESULTS

A.1. Composite Naming Requirements Ballot Response Summary

Requirement	**DCE 1.1**					**DCE 2.0**				
	A	**B**	**C**	**D**	**Tot**	**A**	**B**	**C**	**D**	**Tot**
Seamless XDS API	5	6	3	0	30	7	6	2	0	35
X.500 schema semantics in CDS	0	5	4	4	-2	3	4	4	2	13
X.500 style typed names in CDS	3	0	6	3	3	3	2	5	3	6
Single naming interface	2	6	4	2	14	10	4	0	1	34
Single client protocol for access	2	5	6	2	14	7	3	5	1	28
Access to CDS from non-DCE X.500	0	1	10	1	8	1	8	4	1	19
Access to X.500 using DCE identity	4	4	6	1	22	8	2	3	0	31
Writable browser	1	4	4	4	-1	5	3	5	2	18
Single consistent browser access	0	3	9	2	7	6	4	5	1	27
CDS server export multiple i/fs	1	6	6	1	17	2	5	4	1	16
DME integration for management ops	0	0	5	2	-3	12	5	1	0	47
Add support for hierarchical cells	6	3	2	2	18	6	5	1	1	25
Internationalization enhancements	0	1	7	3	-3	1	1	7	1	8
Code cleanup	9	4	1	0	36	10	2	0	0	34
Fix CDS EBCDIC/ASCII interop	5	1	5	1	18	5	3	2	1	19
Administrative improvements	5	4	6	0	29	7	5	2	0	33
1992 X.500 standards compliance	1	5	7	0	20	7	6	1	0	34
Overall Naming Rating	2	7	6	0	26	8	6	1	0	37

Table 2. Composite Naming Requirements Ballot Response Summary

A.2. System Vendor Ballot Response Summary

Requirement	DCE 1.1					DCE 2.0				
	A	B	C	D	Tot	A	B	C	D	Tot
Seamless XDS API	3	1	3	0	14	3	5	1	0	20
X.500 schema semantics in CDS	0	2	2	3	-6	1	2	3	2	2
X.500 style typed names in CDS	1	0	3	3	-6	1	1	3	3	-4
Single naming interface	0	6	2	1	10	6	2	0	1	18
Single client protocol for access	1	4	3	1	10	4	2	2	1	14
Access to CDS from non-DCE X.500	0	0	7	1	3	0	4	3	1	7
Access to X.500 using DCE identity	2	2	4	0	14	4	2	1	0	17
Writable browser	0	3	2	3	-4	2	2	3	2	5
Single consistent browser access	0	2	4	2	0	3	1	4	1	11
CDS server export multiple i/fs	0	3	3	1	5	1	3	2	1	7
DME integration for management ops	0	0	3	1	-1	5	3	1	0	22
Add support for hierarchical cells	3	1	1	2	4	3	3	0	1	11
Internationalization enhancements	0	1	2	3	-8	1	1	3	1	4
Code cleanup	7	2	0	0	25	7	1	0	0	23
Fix CDS EBCDIC/ASCII interop	3	0	3	1	8	3	1	1	1	8
Administrative improvements	3	2	3	0	16	4	2	1	0	17
1992 X.500 standards compliance	0	3	4	0	10	5	4	0	0	23
Overall Naming Rating	1	3	5	0	14	4	4	0	0	20

Table 3. System Vendor Ballot Response Summary

A.3. End User Ballot Response Summary

Requirement	DCE 1.1					DCE 2.0				
	A	B	C	D	Tot	A	B	C	D	Tot
Seamless XDS API	2	3	0	0	12	4	0	1	0	13
X.500 schema semantics in CDS	0	3	1	1	3	2	2	0	0	10
X.500 style typed names in CDS	2	0	2	2	8	2	1	1	1	9
Single naming interface	2	0	2	0	8	3	1	0	0	11
Single client protocol for access	1	1	3	0	8	2	1	2	0	10
Access to CDS from non-DCE X.500	0	0	3	0	3	1	3	1	0	10
Access to X.500 using DCE identity	2	2	2	0	12	4	0	1	0	13
Writable browser	1	1	2	0	7	2	1	1	0	9
Single consistent browser access	0	1	4	0	6	2	2	1	0	11
CDS server export multiple i/fs	0	2	3	0	7	1	1	2	0	7
DME integration for management ops	0	0	2	1	-2	5	2	0	0	19
Add support for hierarchical cells	3	1	1	0	12	3	1	1	0	12
Internationalization enhancements	0	0	4	0	4	0	0	3	0	3
Code cleanup	1	2	1	0	8	2	1	0	0	8
Fix CDS EBCDIC/ASCII interop	2	1	1	0	9	2	2	0	0	10
Administrative improvements	2	2	2	0	12	2	2	1	0	11
1992 X.500 standards compliance	1	2	2	0	9	2	1	1	0	9
Overall Naming Rating	1	3	0	0	9	3	0	0	0	13

Table 4. End User Ballot Response Summary

A.4. ISV Ballot Response Summary

Requirement	DCE 1.1					DCE 2.0				
	A	B	C	D	Tot	A	B	C	D	Tot
Seamless XDS API	0	2	0	0	4	0	1	0	0	2
X.500 schema semantics in CDS	0	0	1	0	1	0	0	1	0	1
X.500 style typed names in CDS	0	0	1	0	1	0	0	1	0	1
Single naming interface	0	0	0	1	-4	1	1	0	0	5
Single client protocol for access	0	0	0	1	-4	1	0	1	0	4
Access to CDS from non-DCE X.500	0	1	0	0	2	0	1	0	0	2
Access to X.500 using DCE identity	0	0	0	1	-4	0	0	1	0	1
Writable browser	0	0	0	1	-4	1	0	1	0	4
Single consistent browser access	0	0	1	0	1	1	1	0	0	5
CDS server export multiple i/fs	1	1	0	0	5	0	1	0	0	2
DME integration for management ops	0	0	0	0	0	2	0	0	0	6
Add support for hierarchical cells	0	1	0	0	2	0	1	0	0	2
Internationalization enhancements	0	0	1	0	1	0	0	1	0	1
Code cleanup	1	0	0	0	3	1	0	0	0	3
Fix CDS EBCDIC/ASCII interop	0	0	1	0	1	0	0	1	0	1
Administrative improvements	0	0	1	0	1	1	1	0	0	5
1992 X.500 standards compliance	0	0	1	0	1	0	1	0	0	2
Overall Naming Rating	0	1	1	0	3	1	0	1	0	4

Table 5. ISV Ballot Response Summary

A.5. Comments on Ballot Items

In the following comments, the ballot items are identified by alphabetic code (*a* through *q*) as follows:

(a) Seamless XDS API
(b) X.500 schema semantics in CDS
(c) X.500 style typed names in CDS
(d) Single naming interface
(e) Single client protocol for access
(f) Access to CDS from non-DCE X.500
(g) Access to X.500 using DCE identity
(h) Writable browser
(i) Single consistent browser access
(j) CDS server export multiple i/fs
(k) DME integration for management ops
(l) Add support for hierarchical cells
(m) Internationalization enhancements
(n) Code cleanup
(o) Fix CDS EBCDIC/ASCII interop
(p) Administrative improvements
(q) 1992 X.500 standards compliance

Requirement *Comment*

New *(BULL)* Extensibility of GDS to user-defined syntaxes: X.500 has been defined to manage user information. New applications would be candidates to store specific information (new syntaxes) in X.500. Example of applications: Mail systems, EDI, system management, etc. GDS should be opened in order to accept user-extensions to manage these user-defined syntaxes. Quotation: B in V1.1, A in V2.0.

New *(BULL)* Support of MHS package in XDS/XOM/GDS: The syntaxes for supporting MHS in X.500 are defined in the XOM standard. It should be added in V1.1.

New *(BULL)* Support of both Relational Database and C-ISAM in GDS/DSA. A ANSI SQL interface for the DIB should be supported.

New *(IBM)* The DCE 1.0 X.500 implementation supports only RDN's (Relative Distinguished Names) consisting of a single AVA (attribute value assertion) i.e., directory entries with a single distinguished value although the ISO/CCITT standards allow RDN's with multiple AVA's (e.g., <OU=Sales,L=Ipswich> as the RDN of an entry). Some administrations may require using systems to support this in order to interoperate and may stipulate this for conformance testing purposes. This item proposes upgrading the GDS component to support RDN's with multiple AVA's to maximize interoperability with other X.500 implementations.

New *(IBM)* The profiles are OIW, EWOS, AOW, US GOSIP and UK GOSIP.

It is mandatory for an X.500 product (a part of the DCE technology) to claim conformance to the above profiles. They define the communication protocols for vendor interoperability and user information exchange. Being conformed to the profiles will facilitate an X.500 product to interoperate in the open system environment, inside and outside the context of DCE.

OIW, EWOS and AOW — the US, European and Japanese OSI implementors workshops, respectively. These profiles describe the agreements of the X.500 protocols among the participants (including vendors and users) for use in product and test suite development.

GOSIP — Government OSI Profiles. They provide a specification for reference in the procurement of X.500 products for the governments. They define a set of data communication protocols for vendor interoperability and user information exchange. They are based on the Standards and the OIW, EWOS and AOW agreements.

New *(Siemens)* Add the following additional naming enhancement: Add support for the XDS extensions for X.400 (the MHS Directory User Package). We rank this B, A.

General *(UMASS)* The lack of consistent, reliable naming and security interfaces maybe DCE's biggest weakness in 1.0, as much as possible should be done in 1.1, 1.2 and it MUST be all smoothly integrated by 2.0 at the latest.

General *(Boeing)* Whenever a question of doing an enhancement deals with standards, it is our position that it must be the highest priority of any item in the group, e.g., within 2. dce code cleanup, item c., standards compliance is an a priority.

General *(BellCore)* Should support trading functionality (e.g., ODP trader) in addition to the directory functionality.

General *(Apple)* Note that the ''area'' ratings are not an average of the items within. Here, for example, we rate the area high because we rate one of the items very high.

General *(Apple)* Although individual items in this area are more or less important, the overall effect seems to be to move CDS towards an X.500 clone. We think this is the wrong direction. If X.500 is the right metaphor, why not use X.500? We want a simpler model for the local (i.e., cell) level, and don't want X.500 there.

a & b *(BellCore)* CDS should act as a DSA in addition to supporting the GDS interface or the should be direct access to X.500 DSA.

a *(IBM)* The DCE directory services API needs to support the full X/Open Dir. Services API including asynchronous operations.

b *(IBM)* This rating applies to that portion of the X.500 schema model that defines entry classes and the attributes which they must or may have. It does NOT include the Name Bindings portion of the the X.500 schema definition (see comment below). The importance of this is to provide consistent behavior of directory entries between the X.500 GDS and the CDS.

b *(BULL)* It should be investigated if X.500 (GDS) should be the solution instead re-developing a complete new X.500.

b & c *(IBM)* For both of these items, CDS is interpreted to mean the Directory Service that is incorporated with the DCE Security and RPC mechanisms as opposed to implying the current CDS implementation. The reason for doing this is to attempt to separate functional requirements from implementation approaches (i.e., one could meet these requirements by either enhancing the existing CDS implementation or by modifying the current X.500 implementation).

Also OSF should clearly define their position with respect to the relationship between X.500 and the CDS. Is it their intent to develop the CDS into an X.500 look-a-like or to continue to develop it as a separate model for providing directory services? The position on this issue will have a significant impact on the decision as to which implementation should serve as the base for subsequent Directory enhancements.

c *(IBM)* This item is given such a low rating because of the complexity and overhead that implementing a truly compliant version of X.500 naming brings. The Cell Directory Service was implemented as something different than X.500 in the first place for the explicit purpose of being lighter weight and simpler. Trying to reintroduce these semantics would result in losing most of these benefits.

d *(Transarc)* This requires a protocol change. We don't think there should be any protocol changes at this time.

e *(Transarc)* This requires a protocol change. We don't think there should be any protocol changes at this time.

e *(BULL)* A possible solution is to develop remote API's for XDS an XOM.

g *(IBM)* The rationale for this being an <A> is that having to maintain multiple user definitions would be an inhibitor to the acceptance of X.500 as the GDS of choice.

g *(Transarc)* This will make the DCE X.500 implementation incompatible with other X.500 implementations.

g *(Siemens)* This requires 9a (common security API) and probably 9f (delegation).

h & i *(IBM)* OSF should not be spending a great deal of resources on user interface applications as these provide an excellent opportunity for value add extensions and creativity.

j *(IBM)* In order to do this properly, items k (Common interface for name service integration) and n (Name service management) from the ''Integration of DCE into OSF/1'' section of the requirements ballot must also be addressed as they are same problem but different approaches. It is a high priority item to look at the approaches and determine the right way to simplify the migration from other name services to the DCE naming environment.

k *(Siemens)* This item is a subset of what is proposed in item 5. It should be removed from the survey.

l *(IBM)* This prioritization applies only to the ability to address cross cell without a GDA. The security part is addressed in 9e.

l *(Apple)* We feel this is critical to efficient support of multi-cell interaction.

l *(Transarc)* We are in favor of this, but have concerns about security issues.

n *(IBM)* <A> particularly applies to the full test coverage and the specs/code/validation suite consistency items. Complete internal design specifications would be a <B>.

o *(Transarc)* Mainframes could themselves manage the translation from ASCII to EBCDIC.

p *(IBM)* <A> mostly for the Automated Backup subitem. Other two items would be <B>.

q *(IBM)* In the DCE 1.1 timeframe, it is not clear that the 1992 standard will be complete enough to incorporate as a whole into DCE. Those aspects of the 1992 standard that are stable enough should be included, but the DCE 1.1 release of DCE should not be held up in order to have complete compliance.

q *(Apple)* Since it isn't crystal clear, we mention that we assume this is for GDS (X.500) only.

APPENDIX B. DEFERRED REQUIREMENT DESCRIPTIONS

B.1. REQT: Support for X.500 schema semantics within the CDS

GDS ensures the integrity of the Directory Information Database (DIB) by enforcing a set of rules and constraints (defined in the directory schema concerning the structure of DIT, object class definitions, attribute types and their syntaxes, etc.) which are used to represent information in DIB. Every directory entry stored in GDS contains an object class attribute which defines a set of mandatory and optional attributes, naming attributes for the entry and the type of its parent entry etc. Any attempt to modify an entry that would violate the entry's object class or modify the DIT in such a way as to violate the appropriate structure rules fails.

CDS does not have such a schema concept; however, it does define a class attribute which can be used to specify the class of each object entry. For each value of the class attribute, users can define a set of class-specific attributes that can/should belong to that class. However, such information is not stored in CDS and it is left to the users to describe, publish, and inform other users about the characteristics of that object class. In other words, CDS has no knowledge of the object classes and thus cannot enforce any of the rules mentioned above for the class-specific attributes in the object entries. For example, CDS cannot prevent an XDS programmer (assuming appropriate access rights) to delete mandatory attribute RPC_TOWER from an object entry of type RPC_Entry which contains the bindings exported by RPC Server. The outcome of this deletion is that the server will become unreachable from all clients.

B.2. REQT: Support X.500 style typed names within the CDS

In DCE 1.0, the CDS supports passing X.500 style typed names (<attribute type> = <attribute value>) as the cell name portion of a global name. However, the CDS implementation does not support the concept of matching rules that X.500 defines for comparing two names, nor does it support the semantics implied by attribute based naming. Within the scope of the CDS, names are treated as labels rather than Attribute Value Assertions. This requirement would extend the CDS to provide support for the full semantics of X.500 attribute based naming.

B.3. REQT: Support for full X.500 style search within the CDS

In DCE 1.0, the CDS does not support the ''lookup by attributes'' operation functionality described by the X.500 Search operation. Therefore, any objects whose names are within the scope of the CDS cannot be found via the attribute based search operation supported by the XDS programming interface. This requirement would extend the CDS to support the full X.500 style search operation and therefore allow objects within the CDS to be located by attribute filters.

B.4. REQT: Single client protocol for access to naming function

The current DCE implementation requires two protocol stacks in the client workstation in order to support the XDS programming interface. When a CDS name is passed to XDS, it calls the CDS clerk which uses its RPC based protocol to communication with CDS servers to resolve the name. When an X.500 name is passed to XDS, it calls the X.500 DUA and accompanying OSI protocol stack (layers 5, 6, and 7) to communicate with DSAs to resolve the name. Requiring both of these protocol stacks, both of which are fairly large, to reside in every workstation imposes a significant drain on system resources. This is particularly painful in low-end workstation configurations where memory and swap space are at a premium.

Therefore, since RPC is already required on every workstation in order to support the other DCE services, this item proposes modifying the XDS implementation to run over an RPC based protocol and provide access to X.500 via RPC to DAP/DSP protocol gateways.

B.5. REQT: Access to cell name service entries from non-DCE X.500

In DCE 1.0, programs using the DCE XDS API can access names both in any standard X.500 namespace as well as names residing in the CDS. This is accomplished via the switch mechanism supported by the specific DCE XDS implementation. Non-DCE X.500 implementations do not have this capability and therefore cannot provide their users with access to names which reside in a cell namespace. Since the XDS is a standard API, this limits the portability of applications written to XDS which require access to information residing only within a CDS.

This requirement would allow non-DCE X.500 implementations of XDS using X.500 standard DAP and DSP protocols to access and manipulate entries in the CDS subject to access controls.

B.6. REQT: Writable browser

The current CDS browser only provides the capability to graphically display and navigate through a cell name space and its entries. Changes to the namespace or the contents of entries must be made using the text-based CDS control program. This requirement proposes the enhancement of the CDS browser to support all of the CDS functionality through the graphical interface.

B.7. REQT: Single consistent browser across all name exporters

The CDS Browser currently supports only browsing and navigating through cell namespaces. The user/administrator must use other non-graphical tools when accessing resources in other namespaces supported by the DCE (i.e., X.500, DFS, Registry). This requirement proposes enhancing the CDS browser or providing a new generic namespace browser which allows display, navigation and basic attribute manipulation functionality across the entire global namespace.

Note: Providing a truly generic and extensible namespace browser basically is dependent on the implementation of a ‘‘single naming interface across all name exporters’’ requirement. However, a browser which encompasses only the resource managers shipped with DCE is possible without it.

B.8. REQT: Allow CDS server to export multiple name service interfaces

While DCE provides a new, highly functional and integrated set of directory services, there are several existing name services which will continue to exist in parallel with the DCE. For example, the Internet Domain Name Service (DNS) is and will continue to be heavily used by both DCE and non- DCE workstations by e-mail and other existing applications. In an environment where users want to run applications which make use of these existing name services as well as DCE function, the administrator will be required to maintain and keep in synch multiple name service databases containing the same or similar information. For example, in an environment where both DNS and DCE are used, each host must have an entry in both the DCE CDS and DNS. If the host’s IP address is changed, both the CDS and DNS databases must be updated to reflect the change.

This requirement proposes enhancing the CDS servers to support queries against its databases using multiple name service protocols (Internet DNS, NIS, etc.). In this environment, a host would be defined once to the CDS with all of the attributes required by both DCE based and DNS based applications. DCE clients would access the attributes using DCE interfaces and protocols and would be able to take advantage of the full functionality of the DCE. The CDS server would also export the DNS protocols so that DNS based applications, could use existing interfaces to access the same information.

B.9. REQT: Internationalization enhancements

In addition to the general Internationalization requirements for DCE, this item proposes a naming specific enhancement to the DCE naming architecture to allow users to define names for entries in multiple languages so that users accessing those entries can do so via their natural language. For example, a directory in the CDS named `/.:/colors` in US English could be assigned an alternate name `/.:/colores` which would be used when being viewed by a Spanish speaking user.

B.10. REQT: Fix CDS EBCDIC/ASCII interoperability problems

String names for CDS in `cds_types.idl` are defined as an array of type `byte`. This causes problems running in a world where an rpc call is sent to a machine that does not use an ASCII character set. These arrays need to be defined as an array of type `char` so that when the rpc call is received on an EBCDIC machine, the character translations are done correctly.

APPENDIX C. CHANGE HISTORY

C.1. Preliminary Draft

Distributed 1/24/92.

C.2. Draft 2.0

This version incorporated the comments received at the review of the Preliminary Draft held 2/6-2/7, 1992 at the DCE SIG meeting. In addition to minor editorial changes, the following changes were made in this version.

(a) Added this change log.

(b) Modified to support PostScript printers.

(c) Added a description of the pre-1.1 requirements category to the approach section.

(d) Added section on ‘‘Architectural objectives’’ to describe a recommendation on the architectural objectives to drive the evolution of the DCE name services.

(e) The section on ‘‘X.500/CDS relationship’’ has been significantly modified to reflect the discussion at the review and the emphasis on the broader view of strategic recommendations.

(f) The Delivery Recommendations Table has been updated as follows:

  (i) All requirements now refer to the section in which they are described.

  (ii) In the previous version, the table only indicated which ‘‘major’’ requirements were staged across multiple timeframes. In this version, these requirements have been split into specific requirements for each release.

  (iii) ‘‘Single DCE identity (X.500)’’ was moved from 2.0 to 1.1 timeframe.

  (iv) ‘‘Code cleanup’’ has been made a pre-1.1 requirement only since that was the only timeframe unique naming requirements could be identified.

  (v) ‘‘Threadsafe XDS’’ has been split into two requirements, one in the pre-1.1 timeframe for blocking reentrant XDS/XOM libraries. Full X.500 client exploitation of threads has been moved to the 1.1 timeframe.

  (vi) Added requirement on ‘‘Support for hierarchical cells’’ because it was missing from previous draft.

  (vii) Added ‘‘Specifications for major DCE 2.0 requirements’’to 1.1 timeframe. This replaces the use of asterisks in the table to identify those 2.0 work items which need to have design work started in the 1.1 timeframe. Those items are now explicitly identified in the detailed description of this 1.1 requirement.

(g) In the section on ‘‘Seamless XDS API to CDS and X.500’’, removed the reference to search (which now appears as a new deferred requirement in Appendix A) to reflect the view of ‘‘seamlessness’’ defined in the ‘‘Architectural objectives’’ section.

(h) The section on ‘‘Blocking reentrant XDS/XOM libraries’’ has been rewritten to reflect the staging of threadsafe support in the GDS client.

(i) Text has been added to section on ‘‘Decouple XDS from GDS’’ to be more specific about the required configurations to be supported.

(j) The ‘‘Code cleanup’’ requirement has been pared down to address only the naming component specific work items and has been moved up to the pre-1.1 timeframe.

(k) Added ‘‘Exploit threads in X.500 client libraries’’ requirement to 1.1 timeframe requirements.

(l) In the section on ‘‘CDS administrative improvements’’, Item #3 (‘‘Automated backup and restore for the clearinghouse databases’’ has been removed. The current CDS provides the support being requested as best as we could understand the requirement. Also added clarification of the atomicity and online requirements for the bulk operations.

(m) Added section on ‘‘Support for string names at the XDS/XOM API’’ to 1.1 section to reflect the new requirement. Mark Fox is writing the description.

(n) Move ‘‘Access to X.500 using DCE identity’’ requirement from the 2.0 section to the 1.1 section to reflect the updated priority as per the review.

(o) Removed the ''Code cleanup'' and ''Seamless XDS API to CDS and X.500'' sections for the 2.0 requirements descriptions since there were no specific requirements for those items identified in that timeframe.

C.3. Draft 2.1 — Distribution Draft

This was a minor update which complete the set of changes originally targeted for Draft 2.0.

(a) Removed all ''notes to reviewers''.

(b) Completed the Architectural Objectives section for Composite Naming. This is a considerably scaled back version of the section from what was originally planned as was discussed at the April SIG meeting. There was general agreement that any more detail in this document at this time was not required given the amount of longer term architectural work remaining to be done and the lack of time to focus on it.

(c) Added the Minimum Search Subset requirement proposed by IBM Toronto at the April working meeting as a 1.1 requirement (see the requirement on ''Minimal XDS Search Capability for CDS'').

(d) Added clarification text to the X.500/CDS Relationship section to clarify the question of ''integration of X.500 into the cell''.

(e) Added detailed description to the requirement on ''Support for string names at the XDS/XOM API''.

(f) Added a brief example to the requirement on ''Single naming interface''.

C.4. DCE-RFC Version

Published as DCE-RFC 4.0 in June 1992.

APPENDIX D. DCE NAMING WORKING GROUP MEMBERSHIP

alm@apple.com	Ann McLaughlin (Apple)	408-974-5540
amal@ausvmq.vnet.ibm.com	Amal-Shahee-Gouda (IBM)	512-838-2196
arnet@cup.hp.com	Arne Thormadsen (HP)	408-447-4798
art@cs.umass.edu	Art Gaylord (U. Mass)	413-545-2520
bank@pokvmcr3.vnet.ibm.com	Judi Bank (IBM)	814-296-5350
c.j.lambert.ste0902@oasis.icl.co.uk	C.J. Lambert (ICL)	+44-638-72616
cwf@cray.com	Charles Fumuso (Cray)	612-683-5449
emartin@apollo.hp.com	Liza Martin (HP)	508-256-6600
hanlon@apollo.hp.com	Bart Hanlon (HP)	508-256-6600
helene@osf.org	Helene Hurot (Bull)	617-621-7282
jackson@took.enet.dec.com	Jim Jackson (DEC)	508-486-5272
kwok@torolab5.vnet.ibm.com	Ray Kwok (IBM)	416-448-3032
m_fox@took.enet.dec.com	Mark Fox (DEC)	508-486-5577
melman@osf.org	Howard Melman (OSF)	617-621-8989
nichols@took.enet.dec.com	Wick Nichols (DEC)	508-486-5266
nl@osf.org	Norbert Leser (OSF)	617-621-8715
ohara@sni-usa.com	Abigail O'Hara (SNI)	617-349-5016
printis@parc.xerox.com	Bob Printis (Xerox)	415-812-4465
renand@citi.umich.edu	Brian Renand (U of M)	313-936-2481
roc@apollo.hp.com	Richard Curtis (HP)	508-256-6600
rose_1@apollo.com	Larry Rose (HP)	508-256-6600
ssnyder@ausvm1.vnet.ibm.com	Scott Snyder (IBM)	512-838-8134
staylor@afterlife.ncsc.mil	Steve Taylor (DoD)	301-859-6695
steve@locus.com	Steve Kiser (Locus)	310-337-5261
stokes@ejstokes.austin.ibm.com	Ellen Stokes (IBM)	512-838-3725
tennent@espresso.boeing.com	Dick Tennent (Boeing)	206-805-3613

Table 6. DCE Naming Working Group Membership

REFERENCES

[Mart 91] L. Martin, ‘‘Directory Services and Naming’’, paper submitted to the DCE SIG, March 8, 1991.

[OSF 90a] OSF, ‘‘Distributed Computing Environment Rationale’’, White Paper, May 15, 1990.

[OSF 90b] OSF, ‘‘Directory Services for a Distributed Computing Environment’’, White Paper, September, 1900.

[OSF 90c] OSF, ‘‘Distributed Computing Environment, An Overview’’, White Paper, November, 1990.

AUTHOR’S ADDRESS

Scott Snyder
IBM Austin
11400 Burnet Road
Internal Mailstop #9641
Austin, TX 78758
USA

Internet email: ssnyder@ausvm1.vnet.ibm.com
Telephone: +1-512-838-8134