OSF DCE SIG **H. Melman (OSF)**
Request For Comments: 35.0 **January 1993**

LOCATION OF INSTALLED DCE FILES

1. INTRODUCTION

This is the guideline for locating the DCE files (executables, libraries, configuration files, etc.) on target client and server systems for DCE 1.0. DCE has no architectual dependencies upon this guideline. It should be considered as a recommendation to venders and as a description of the default reference implementation configuration.

2. GOALS

The main purpose for (re)structuring the location of DCE files is to provide a uniform model for supporting administrators, application programmers, and users of DCE components. A minimum of configuration effort should be required to install and run DCE on various platforms.

Other objectives are:

(a) Accommodating conventional centralized file organizations (i.e., Unix systems such as OSF/1 and SVR4 — see Appendix A) by utilizing DCE DFS facilities in the cell wide environment.

(b) Uniformity across different system architectures by separating architecture independent, architecture dependent and machine private files in respective subtrees.

(c) Portability to non-Unix environments, although the model is derived and implemented on Unix based systems.

(d) Separating areas which require frequent daily write operations from read only areas.

(e) Privileged DCE components should be able to find their databases without being spoofed by a malicious user.

(f) Utilizing DCE cell-wide file access where available (i.e., DFS).

(g) Collecting DCE files in distinct hierarchies, separated from the base operating system organization.

(h) Removing absolute path name references from the documentation (i.e., administration guides and manual pages) and include instructions only in the *DCE Porting and Testing Guide* [PORTG], *DCE Administration Guide* [ADMIN] and the *DCE Release Notes* [RELN].

(i) Allow for multiple versions concurrently operating inside cells, only associate local systems (machines) to a particular DCE release. Support for installation, de-installation, and upgrade procedures.

3. DCE SUBTREES

This leads into three distinct sets which need to be combined in organizing the DCE file locations. These are listed in descending order, which means that it's preferable for several reasons (such as easier upgrading and de-installing of DCE) to include as many files as sensible into the first tree structure.

3.1. dceshared

For one set of files it's appropriate to create a top-level branch into DCE in the form <path-from-root>/<product-name>/<subdirectory(ies)>/* (e.g., `/opt/dce1.0/bin/vos`). This set can either be kept on local systems or preferably exported to the DCE cell via DFS. Therefore, sharable files including binaries (addressed via `@sys`) are stored here. *dceshared* is a read-only subtree.

In fact, all files generated by a DCE build, all files delivered to binary licensees, and, if appropriate, all files delivered to source licensees will be initially stored, kept and upgraded in this location. If necessary, copies or symbolic links to the other locations will be generated.

> Note: With the exception of *dceshared*/etc, all files in *dceshared* will be kept unmodified over the lifetime of an installed DCE release. If necessary, configuration and data files are only stored as templates in this location. The actual working set of data files are located below *dcelocal*/var, *dcelocal*/etc, and *dceshared*/etc.

Although the DCE Executive and various server packages have to be licensed separately to binary licensees, this guideline is based on the assumption that one wants to utilize the DCE capability of sharing files cell wide. Storing and managing these files in a central place and running certain servers on dedicated server systems are two distinct tasks.

> Note: DCE object files, as they will be shipped to end-users are separated in different packages (Executive and various Servers). Each of these licensed packages are self-contained, which means they include the complete set of required files. The Executive, for example, contains all utilities for application programmers including DCE header files and libdce.a. Since users in DFS cells may not want to have replicas of these files physically stored on their local system, they can remove their locally installed *dceshared* tree and redirect the default symbolic link to the cell wide *dceshared* (see below), if these particular files are available there. This applies similarly to the Server packages.

For the capability of initially booting and configuring the cell (i.e., Big Bang), the appropriate files for mandatory servers (CDS and Security) have to be available on that server machine (see *dcelocal*), but files for the optional servers (e.g., GDS, LM Server) may reside in remote filesets, dependent on the configuration (i.e., availability of DFS). Nevertheless, we strongly recommend that server systems should keep copies of the minimal set of tools and data files local for the capability of stand-alone operation and emergency maintenance (see *dcelocal*).

dceshared is by default /opt/dce, which in turn is, on newly initialized systems, a symbolic link to /opt/dce1.0. After configuring the system into the DFS based cell environment, this symbolic link usually points to /:/opt/dce1.0 (short form of /.:/fs/opt/dce1.0) — see also below. This accommodates the requirement for the initial cell set-up phase (i.e., Big Bang), where *dceshared* must be available on that local system.

The pointer of *dceshared* has a version number associated with the path name. This provides the capability of running multiple DCE versions in one cell which may be required in an intermediate phase of upgrading to a new release. An additional advantage would be a cleaner de-installation procedure. In case of upgrades to a newer release, configuration files which are kept in *dceshared*/etc need to be transferred to the new subtree and linked back to the old release.

3.2. dcelocal

dcelocal serves two purposes:

(a) It is necessary that a small set of DCE files be on the local disk to start up the various DCE components, for local configuration, and for log info. Except for diskless support, none of these files have to be available in the root partition, since DCE initialization takes place after mounting the local file systems. If diskless is supported, very few files (e.g., dfsd) must have copies in the root partition (i.e., /sbin).

(b) The appropriate files on DCE servers which have to be local to the server station must be stored under *dcelocal*. All server specific data files are located below *dcelocal*/var/*dce-component-name*.

The default path for *dcelocal* is set to /opt/dcelocal. This is a fixed path name eventually determined and documented by vendors.

The contents of *dcelocal* may vary from machine to machine inside a DCE cell. Every machine must have local access to files for running the system stand-alone (if disconnected or partitioned from cell), and must have local access to those files which are necessary to initialize it as DCE client station up to activating DFS access in the cell.

> Pitfall: Administrators have to be aware that during DCE initialization only executables in *dcelocal*/bin are reliably available. Therefore, startup procedures such as rc-scripts should address executables through *dcelocal* rather than /usr/bin, even if these same files supposed to be there.

There is no version number attached to *dcelocal*, since multiple DCE versions are not supported on single machines and servers are generally capable of serving previous releases.

3.3. Conventional Unix Locations

A small set of DCE files and directories should be accessible in conventional locations for:

(a) user convenience for frequently used utilities such as `idl` in `usr/bin` and `localtime` in `/etc/zoneinfo`.

(b) DCE application programmers' requirements who expect header files in `/usr/include` (or a subdirectory thereof, i.e., `/usr/include/dce`) and libraries in `/usr/lib` (i.e., `libdce.a`).

The intention is to store the complete set of delivered DCE files (except those which are created during runtime) in the first two subtree structures. Files which are required in conventional Unix locations and executables in *dcelocal* may be duplicates of files and templates in *dceshared*. Vendors will eventually decide which set of files they want to be available in their ‘‘standard’’ locations (such as `/usr/bin`) and whether they want them copied or just created as symbolic links.

4. DEFAULT SETTINGS AT COMPILE-TIME

For convenience, and due to the necessity of supporting traditional Unix file system organizations as well as non-Unix systems, we need to introduce a mechanism which allows us to point to the top of the product-specific DCE trees (i.e., *dceshared* and *dcelocal* — see discussion above). A compile rather than a run time selection is necessary for security reasons, which excludes the use of environment variables.

The introduction of a separate configuration file provides enough flexibility and protection for this purpose. Nevertheless, there are two issues which cannot completely be solved by using this approach:

(a) To support multiple DCE releases per cell, a version number would always be associated with the path name. If not using yet another level of indirection, the symbolic links - for example - created for files such as `/usr/bin/idl` to *dceshared*`/bin/idl` would have to have this version number in the path name, requiring additional reconfiguration work while upgrading DCE (i.e., redirecting all symbolic links to the new release).

(b) Since DCE binary products contain the whole set of necessary files (i.e., files in *dceshared* and in *dcelocal*) one may want to redirect to the cell wide available *dceshared* and delete the local copies. Again, this would require changing all symbolic links (similar to issue (a)). The same effort would be required after the ‘‘Big Bang’’ configuration.

The preferable and much more elegant technique would be the use of symbolic links (in Unix) for addressing the locations *dceshared*. There is no such requirement for *dcelocal*, since it can be determined at compile time which path on the local system ought to be used. Given the right access settings on the directory containing the symbolic link (write for root or administrator, read for other users), the protection is equivalent to using a configuration file. Most of the other (non-Unix) operating systems provide techniques similar to symbolic links (e.g., aliases, symbolic names).

An additional advantage of this approach would be the avoidance of a separate configuration file and respective lookups during run time via a function call (i.e., `getdcepath()`). Traversing a symbolic link which is kept on the local system should be slightly faster than opening and parsing a file.

The path name for the entry containing the symbolic link will be passed to the DCE sources during compile time. It will be implemented in setting `-D` flags on compiler command lines:[1]

```
-DDCESHARED_PATH="/opt/dce" -DDCELOCAL_PATH="/opt/dcelocal"
```

1. This method was used for DCE 1.0, it is being changed for DCE 1.1. Please see [RFC 34.0] for details on the new mechanism.

These flags will be defined in a central location (`osf.dce.mk`) using the variable `DCEPATHS`. Each individual `Makefile` which requires these values must add `${DCEPATHS}` to `CFLAGS`. Vendors can modify the default entries in `osf.dce.mk` or overwrite DCEPATHS in the environment (e.g., `setenv DCEPATHS "xxx"`). Future DCE releases may use a common include file (i.e., `dce.h`) instead.

The install script for the DCE components will have to create a symbolic link from `/opt/dce` to `/opt/dce1.0` and to create the directory `/opt/dcelocal` with appropriate rights.

This approach is sufficient since system vendors can easily change these few entries to accommodate their system design and requirements (e.g., `/opt/dce` to `/usr/dce`).

Some care must be taken by system administrators who are frequently using files located in *`dceshared`* or *`dcelocal`* and therefore need them to be set in their PATH variable. In using different workstations, this could be different depending on system vendor defaults. Not avoidable for *`dcelocal`*. But even for *`dceshared`*, an administrator or user of DCE usually wants to access the right DCE version for the given workstation rather than the latest DCE release which could be easily accessible via a link from `/:/opt/dce` to `/:/opt/dce[latest]`.

Again, this problem occurs if multivendor systems with different default settings in a given cell are used. A solution certainly would be an additional indirection managed by the respective user using something like `$MACHINE`. However, this needs to be documented in the *DCE Administration Guide*.

5. DOCUMENTATION

DCE utilities and services which expect files in these locations (e.g., `BosConfig` in the `bos` command) must be modified respectively, and references in the documentation (i.e., man pages) must meet this guideline. Instead of using absolute path names, the references in the documentation are relative to the respective roots of DCE file locations. The same notation as used in this document (*`dceshared`* and *`dcelocal`*) for identifying the path prefix is used.

The following documents should contain information about this guideline:

(a) Installation Notes (Part of *DCE Administration Guide*):

Explanation of how to prepare the installation of DCE (e.g., set up `/opt` as separate file system with sufficient space).

(b) ‘‘Big Bang’’ configuration (Part of *DCE Administration Guide*):

Basically elaborating the cases that after initial configuration the access to *`dceshared`* might be reconfigured to utilize DFS access.

(c) *DCE Porting and Testing Guide*:

Porting instructions for setting and compiling the default values.

(d) *DCE Release Notes*:

Status of this proposal, actual directory layout, version numbering etc.

(e) *DCE Administration Guide*:

Explanation of naming conventions in documentation for locations relative to *`dceshared`* and *`dcelocal`*, hints and pitfalls for administrators, probably example configuration scenarios.

6. DIRECTORY LAYOUT

The following directories will be created during DCE installation: [← indicates a symbolic link pointing to the first entry]

All directories are created with Unix permissions set to `rwxr-xr-x` with user `root` and group `bin`. In subsequent configurations, the Security Service might define roles for several administrators (principals or groups). A conceivable scenario could be:

(a) Software Administrator (owner of installed software packages), with write and modification permission for the entire set of files included in *dceshared* (except *dceshared*/`etc/*`).

(b) Cell Administrator, with read and write permission to cell wide configuration and log files.

(c) DCE Service Administrators (responsible for a particular DCE service such as Security), with read and write permission to data files for respective service. In most cases one might want to design a separate Security Service Administrator, while a single Cell Administrator is responsible for the remaining DCE services.

(d) Local DCE System Administrator (responsible for client setups), with read and write permission to respective local files.

The directory structure supports such a model which can be easily managed if ACLs are available. Specific files might require different access permissions to be determined by component suppliers.

(a) *dceshared* will be created as `/opt/dce` which is a symbolic link (`/opt/dce1.0` ← `/opt/dce`)

This is an entry always physically located at the local machine, and therefore the Local DCE System Administrator (or the respective Software Administrator) must have write permission to modify this link. The general procedure would be to redirect this link from the local to the cell wide accessible fileset as soon as the local system is configured and the cell is available.

> Note: Since this link is crucial for protecting the local station if running as client and the server station if acting as service requester, special care must be taken. This symbolic link must be created in a protected directory (only Local System Administrator have write and modification permission). If this cannot be guaranteed with `/opt`, `/etc` would be an alternative.

(b) *dceshared*/`bin`

Utilities for applications programmers and DCE users, DCE administration utilities, server processes (daemons).

(c) *dceshared*/`etc`

Templates of configuration files etc., which are in architecture dependent format.

> Note: In accordance with the efforts at OSF and AT&T of moving traditional administration utilities out of `/etc`, we should avoid storing any executables in any of the outlined `etc` directories. Only configuration files and administration databases belong in `etc` directories.

(d) *dceshared*/`etc/zoneinfo` (Write permission for Cell Admin)

Templates of respective configuration tables.

(e) *dceshared*/`examples`

Example and test executable files

(f) *dceshared*/`lib`

Application libraries (`libdce.a`) and DCE internal libraries.

> Note: If there are libraries remaining, which are only necessary for building DCE itself, they should go into this location.

(g) *dceshared*/`nls/msg/${LANG}`

Contains the delivered message catalogues (`*.cat`) files for each supported language.

(h) *dceshared*/`share/doc`

Complete set of DCE documentation as organized right now.

(i) *dceshared*/`share/etc`

Templates of common (shared) configuration files.

(j) `dceshared/share/include` ← `/usr/include/dce`

Application header files and DCE internal header files. This entire directory is linked to `/usr/include/dce`, but in future DCE releases we might want to separate and link only those files which are necessary for writing DCE based applications.

> Note: Stub files should not be exported (the binaries already exist in `libdce.a`).

(k) `dceshared/share/sources`

DCE sources and build tools as organized in the ODE build tree.

(l) `dcelocal`

directory (default: `/opt/dcelocal`)

(m) `dcelocal/bin`

DCE administration utilities and Server processes (daemons) necessary for local client system initialization and for respective server stations.

(n) `dcelocal/etc` (Write permission for Local System Admin)

Local (machine-private) configuration files maintained by client systems.

(o) `dcelocal/var/adm/dce-component-name` (Write permission for Local System Admin)

Storage of log files (including core images) and cache files, maintained by client systems only. For convenience, symbolic links from `/var/adm/dce/client/dce-component-name` will be created.

> Note: It has been asked why shouldn't servers put caches and logs here as well. That's a good question. Except for the probability of name-clashes (GDS, for example, has the same names but different sub-directories for clients and servers), there's no further technical reason. Mainly, it's for the sake of ''cleanliness''. We want to separate client from server files. In cases, such as deinstallation we basically have to remove the entire server subtree.

(p) `dcelocal/var/dce-component-name` (Write permission for Service Admin)

This subtree contains all data files (configuration files, databases, etc.) which are maintained by each of the DCE servers. To provide high availability and (in case of the Security Service) appropriate protection, data files associated to a service are usually physically located at the server site. Therefore they are stored in separate trees below `dcelocal/var`. If a DCE component provides several services such as fileset location server in DFS, respective substructures may be created (e.g., `file/filesetloc`). Files in `dcelocal/var/dce-component-name` are only those which are accessed by dedicated servers; other data files which are shared among DCE components will be located in `dceshared/etc`.

If one service does not assume the files to be physically local, this organization allows for treating this subtree as exportable fileset, for example.

> Note: Configuration and log files relative to client systems are not stored here; they are in `dcelocal/etc` and `dcelocal/var/adm/dce-component-name`. In general, the subdirectories `dcelocal/var/dce-component-name` are explictly preserved for server usage only (i.e., for databases, config files, logs, ...). Therefore, one will find something under `dcelocal/var/adm/dce-component-name` on every (client) system and `dcelocal/var/dce-component-name` will only be populated on those systems which are running as server for that particular component.

(q) `dcelocal/var/dce-component-name/adm` (Write permission for Service Admin)

This subdirectory should be maintained by each service to store log files (including core images) and cache files generated by servers. Since users may expect log files in conventional locations, `/var/adm/dce/dce-component-name` will be created as symbolic links to these directories.

(r) `/etc/zoneinfo`

Copies of templates in *dceshared*`/etc/zoneinfo`, modified if necessary.

> Note: Since these files may already existent and respectively modified on the local system, the installation procedure must take care not to simply overwrite them!

(s) `/krb5` (Write permission for Local System Admin)

Directory for Kerberos configuration files for conventional Kerberos 5 environment. Symbolic links to appropriate files in *dcelocal*`/etc`.

(t) `/sbin`

Small set of executables required in the root partition, derived and copied from *dceshared*`/bin`.

> Note: Although this is an exact subset of executables to be found in *dcelocal* as well, keeping the copy in *dcelocal*`/bin` is sensible, since on running systems `/sbin` is usually not set in `$PATH`.

(u) `/usr/bin`

Utilities for applications programmers and DCE users. Most of these are symbolic links pointing to *dceshared*`/bin`. On server stations, copies of respective executables may be necessary for the capability of initializing the system.

> Note: Some of these, such as `login` and `su`, may actually be local copies for required high availability.

(v) `/usr/lib`

Symbolic link from `libdce.a` to *dceshared*`/lib/libdce.a`.

(w) `$NLSPATH`

Copies of NLS message catalogues (`*.cat` files) for respective languages.

> Note: Rather than creating symbolic links to respective locations in *dceshared*`/nls/msg/${LANG}`, copies are necessary for performance and availability.

The file organization is set up in such a way that regular users including application programmers do not need (frequent) access to files stored under *dceshared* and *dcelocal*, and therefore don't need to set these in their `$PATH` variable. Usually DCE system administrators require frequent access to these locations.

The contents of subdirectories below *dceshared* correspond to directories conventionally located below `/usr`. As exception, *dceshared*`/etc` corresponds to `/etc`. Similar mappings apply to directories in *dcelocal*. It needs to be pointed out that ''etc'' directories contain data files for administration purposes which are usually read and modified only by permitted administrators for reconfiguration etc. Files below ''var'' directories on the other hand, are typically frequently modified through client and server operations [See also Appendix A].

Since *dcelocal*`/var/`*dce-component-name* could contain large databases, sufficient disk space must be provided for.

APPENDIX A. File System Organizations

This is an example of file system organization provided with OSF/1 as well as with SVR4 (see also `filesystem(7)` man page for SVR4 and ''System Administrator's Guide'' chapter 7.2 for OSF/1).

This organization is based on the assumption of three distinct sets of file types:

(a) Files sharable inside the entire cell (or a logical subset of). These are Text files, whether in human readable form which can be edited with regular text editors (e.g., configuration files, profiles) or in decrypted form such as password strings.

(b) Architecture dependent files which may be shared among similar architectures (same CPU, OS). Such files are executable binaries or libraries. If shared and accessed via DCE DFS, linking these files (preferable subdirectories) with `@sys` is a convenient mechanism.

(c) Machine private files. In this category belong files which are only relevant for a single system (machine) in the DCE environment, and files which only span the lifetime of a particular system. Examples are rc-scripts, log files, and executable binaries which have to be available on systems running in stand-alone mode (boot phase, disconnected from the network).

The example file system organization is as follows:

(a) `/`

Contains all the files necessary for booting the system. Subdirectories are reorganized to accommodate file sharing in networked environments. Only `/sbin`, `/etc`, and `/dev` are subdirectories; the other directories are mount points to other file systems or can be treated as such (i.e., `/tmp`).

(b) `/sbin`

Essential executables for administration and operation required for boot and initialize the system in single user mode. These files are architecture-dependent.

(c) `/etc`

Machine-private administration and configuration files and sys-admin databases (SVR4 has no executables in `/etc` any more, while OSF/1 keeps these as symbolic links to `/usr/sbin`).

(d) `/dev`

As in traditional Unix systems. AT&T reorganized the internal layout for improved manageability.

(e) `/tmp`

As usual. There are no specific guidelines but it could be treated as mount point to something such as `/var/tmp` or `/var/tmp@host`.

(f) `/export`

The default root for the exported file system tree.

(g) `/mnt`

The default temporary mount point for file systems.

(h) `/opt`

The root of a subtree for add-on applications packages. SVR4 has introduced `/opt`, but a detailed plan hasn't been worked out yet.

(i) `/proc`

The root of the process file system (in SVR4).

(j) `/usr`

Contains sharable files that are static over the life of the system. Except under `/usr/share`, the files and dirs in `/usr` (such as `/usr/bin`) are architecture (i.e., CPU and OS) dependent. System-modified files have been moved to `/var`. OSF/1 installs `/home` and `/var` but doesn't provide specific guidelines which leaves those files remain in `/usr`. In OSF/1 (I'm not sure about SVR4), /usr can be a subdirectory with several mounts or a mount point itself to accommodate various needs.

(k) `/home`

Contains the home dirs and the files of users.

(l) `/var`

Files and dirs which change over the life of the local system (log files, mail, spool etc.).

APPENDIX B. Directory/File Layout example

This appendix is as it appeared in the original version of this document. For an updated list of the file organization please see the install tree generated by a DCE build.

Note: The file layouts for DFS and Security are derived from input from Transarc and HP. Others need to be incorporated by technology providers. It is their responsibility to resolve possible name conflicts or other problems.

(a) *dceshared*/bin

```
rpccp, rpcd, idl, nidl_to_idl, uuidgen, dtscp, dtsd, time
provider, cdscp, cdsd, cdsadv, cdsbrowser, cdsclerk,
gdad, gdsadmin, gdsadm, gdscache, gdsdsa, gdsdaemon,
gdsstubs, other gds*, acl_edit, sec_admin, sec_createdb,
sec_salvagedb, passwd_import, passwd_refresh, kinit,
klist, kdestroy, chpass, login, su, fts, cm, cell, bos,
bak, scout, dfsd, fxd, bosserver, ftserver, flserver,
repserver, bakserver, upclient, upserver, dfsexport,
newaggr, salvager, mount
```

(b) *dceshared*/etc

(c) *dceshared*/etc/zoneinfo

```
localtime, posixrules
```

(d) *dceshared*/examples

(e) *dceshared*/lib

```
libdce.a
```

(f) *dceshared*/share/doc

(g) *dceshared*/share/etc

(h) *dceshared*/share/include

```
exported DCE *.h, *.idl
```

(i) *dceshared*/share/sources

(j) *dcelocal*/bin

```
bos, fts, cell, dfsd, fxd, bosserver, ftserver, flserver,
repserver, bakserver, upclient, upserver, dfsexport,
newaggr, salvager, mount rpcd, dtsd, secd
```

(k) *dcelocal*/etc

```
cacheinfo,
passwd_override, passwd_override.private, local_registry,
local_registry.private, krb.conf, v5srvtab
```

(l) *dcelocal*/var/adm/*dce-component-name*

   (i) *dcelocal*/var/adm/dfs

   ```
   DFSLog  (old AFSLog)
   ```

   (ii) *dcelocal*/var/adm/directory/gds

   ```
   dua/..., cstub/...
   ```

   (iii) *dcelocal*/var/adm/security

   ```
   error_log
   ```

(m) *dcelocal*/var/*dce-component-name*

(n) *dcelocal*/var/dfs

```
fldb.*, bkdb.*, BosConfig, UserList.bosserver,
UserList.ftserver, UserList.flserver, UserList.repserver,
UserList.bakserver, UserList.upserver, dfstab
```

(o) *dcelocal*/var/directory/cds

```
cdscp.bpt
```

(p) *dcelocal*/var/security

```
rgy_data/...
```

(q) *dcelocal*/var/time

```
dtsscp.bpt
```

(r) *dcelocal*/var/*dce-component-name*/adm

(s) *dcelocal*/var/dfs/adm

```
FileLog, FtLog, RepLog, FlLog, SalvageLog, BosLog,
server core files
```

(t) /etc/zoneinfo

(u) /krb5

Symbolic links to krb.conf and v5srvtab in *dcelocal*/etc

(v) /sbin

```
dfsd, ...
```

(w) /usr/bin

Symbolic links to dedicated files in *dceshared*/bin such as:

```
fts, cm, cell,
idl, uuid_gen, cdsbrowser, acl_edit, login, su, chpass,
kinit, klist, kdestroy
```

(x) /usr/lib

Symbolic link to *dceshared*/lib/libdce.a

(y) $NLSPATH

Copies of `*.cat` files from *dceshared*/nls/msg/${LANG} into the system wide language dependent subdirectories for message catalogues.

C. ACKNOWLEDGEMENTS

This RFC is based on a document originally sent to DCE developers during DCE 1.0 development written by Norbert Leser of OSF and dated May 2, 1991.

REFERENCES

[ADMIN] DCE Administration Guide, Update 1.0.1, Open Software Foundation, 1992.

[PORTG] DCE Porting and Testing Guide, Update 1.0.1, Open Software Foundation, 1992.

[RELN] DCE Release Notes, Update 1.0.1, Open Software Foundation, 1992.

[RFC 34.0] H. Melman, ‘‘DCE Coding Style Guide’’, to appear.

AUTHOR'S ADDRESS

Howard Melman
Open Software Foundation
11 Cambridge Center
Cambridge, MA 02142
USA

Internet email: melman@osf.org
Telephone: +1-617-621-8989