

HP DCE/9000 Application Development Tools for HP-UX 11i Release Note

Version 1.8

HP 9000 Systems



Manufacturing Part Number: B3193-90023

E0301

United States

© Copyright 1983-2000. Hewlett-Packard Company. All rights reserved.

Legal Notices

The information in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty. A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Restricted Rights Legend. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

HEWLETT-PACKARD COMPANY
3000 Hanover Street
Palo Alto, California 94304
U.S.A.

Use of this manual and flexible disk(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

Copyright Notices. ©copyright 1983-95 Hewlett-Packard Company, all rights reserved.

Reproduction, adaptation, or translation of this document without prior written permission is prohibited, except as allowed under the copyright laws.

©copyright 1979, 1980, 1983, 1985-93 Regents of the University of California

This software is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California.

©copyright 1980, 1984, 1986 Novell, Inc.
©copyright 1986-1992 Sun Microsystems, Inc.
©copyright 1985-86, 1988 Massachusetts Institute of Technology.
©copyright 1989-93 The Open Software Foundation, Inc.
©copyright 1986 Digital Equipment Corporation.
©copyright 1990 Motorola, Inc.
©copyright 1990, 1991, 1992 Cornell University
©copyright 1989-1991 The University of Maryland
©copyright 1988 Carnegie Mellon University

Trademark Notices. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X Window System is a trademark of the Massachusetts Institute of Technology.

MS-DOS and Microsoft are U.S. registered trademarks of Microsoft Corporation.

OSF/Motif is a trademark of the Open Software Foundation, Inc. in the U.S. and other countries

1 HP DCE/9000 Version 1.8 Application Development Tools for HP-UX 11i Release Note

This release note contains information for Version 1.8 of HP DCE/9000 Application Development Tools for HP-UX 11i.

Announcement

The following information is for version 1.8 of the HP DCE/9000 product.

What's in This Version

The HP DCE Version 1.8 Application Development Tools for HP-UX 11i help users to develop HP DCE 1.8 applications.

NOTE The previous version of the Release Notes are available in the following directory: `/opt/dce/newconfig/RelNotes`.

Application Development Tools Product Options

The HP DCE Version 1.8 Application Development Tools for HP-UX 11i consists of the following optional products.

The licenses you have purchased from Hewlett-Packard determine whether you can install this product. The optional products are:

- **HP DCE Application Development Tools** assist in the development of the HP DCE programs written in C. The following are the different HP DCE Application Development Tools and their corresponding Man Pages:

HP DCE-C-Tools include:

- ✓ *HP DCE* Version 1.8 Messaging and Serviceability Interfaces
- ✓ i2DL Compiler
- ✓ Sample Applications

DCE-CoreTools.DCE-BPRG include:

- ✓ HP DCE Core Programming Tools

DCE-CoreTools.DCEP-ENG-A-MAN

- ✓ HP DCE Core Programming Manpages

DCE-CoreTools.TH-ENG-A-MAN include:

- ✓ HP DCE Threads Programming Manpages

DCE-CDS-Server.CDSS-ENG-A-MAN include:

- ✓ HP DCE-CDS-Server Manpages

DCE-SEC-Server.SECS-ENG-A-MAN include:

- ✓ Manpages for DCE Security Server Software

NOTE Although Tracing, Logging and Error Reporting facilities of the previous versions are still available in this version of HP-DCE-C Tools, HP recommends that you use the new Messaging and Serviceability Interfaces.

New Features

The following features have been added at HP DCE 1.8:

- 32 / 64 bit DCE Application Development with kernel Threads. HP DCE Version 1.8 core services comes with 32 bit and 64 bit DCE kernel threaded libraries to support Application Development with kernel Threads.

To use HP-UX kernel Threads, DCE applications should either define the pre-compiler name `_POSIX_D10_THREADS` or define `_POSIX_C_SOURCE = 1995061`.

DCE Applications built based on kernel threads, which use exception handling mechanisms must include the header files `<pthread_exc.h>` and `<exc_handling.h>` and include the path `/usr/include/` before `/usr/include/dce` (or `/opt/dce/include/dce`) in the search path for DCE applications using the user-level threads library. However, on the same machine (s) where the kernel Threads Development Tools have been installed users must use `/usr/include/dce/` (or `/opt/dce/include/dce`) before `/usr/include` in the search path for compilation.

Kernel Threads based DCE applications must link with the following libraries

1. `/usr/lib/libdcekt.sl`
2. `/usr/lib/libpthread.sl`

What's in This Version

NOTE

The 64 Bit DCE applications may need to include “/usr/include/dce/dce64.h”. The site “<http://www.software.hp.com/STK>” contains 64 bit porting guidelines, 64 bit compiler and linker changes needed to port the applications to 64 bit. The advantages of moving to 64 bit can be found under

“<http://www.software.hp.com/STK/hpuxoverview.html#64-bit>”.

The 32 Bit version kernel threads DCE library (libdcekt.sl) which was earlier released on AP9806 is now part of the HP-UX base system.

-
- **DCE Library:** Starting HP-UX 11i there will only be one version of DCE library (and dced). 56 bit DES encryption which was earlier restricted to US / Canada is now available to all customers. Hence, there will be no separate DCE-Domestic product in DCE 1.8 version.
 - **Libraries shipped in DCE-KT-Tools:** DCE -KT-SHLIB fileset and DCE-KT-Domestic. DCE-KT-SHLIB are not part of the DCE core.
 - **DCE-KT-Tools:** DCE-KT-BPRG fileset of HP DCE 1.7.1 will be part of DCE-CoreTools product. DCE-KT-Tools product will not be available for HP DCE 1.8
 - **Filesets in IntegratedLogin** has been renamed. Fileset AUTH-COMMON is renamed to ILOGIN-COMMON and AUTH-DCE is renamed to ILOGIN-DCE.
 - The **binaries of HP-DCE kinit, klist and kdestroy** are in directory /opt/dce/bin. The kinit, klist and kdestroy in /usr/bin belong to KRB5-Client product. For details on HP DCE kinit, klist and kdestroy: refer HP DCE kinit, klist and kdestroy manpages found in /opt/dce/share/man directory.

Fixes

See the section, “Patches and Fixes in this Version” for a list of the fixes included in this version.

Manuals Available for this Version

Describes the documentation available for the HP DCE 1.8 Application Development Tools for HP-UX 11i.

Printed Documentation

The printed documentation for the Application Development Tools consists of the following:

- This document describes the Application Development Tools as described in the section ***Application Development Tools Product Options***.
- Planning and Configuring HP DCE 1.8 (B3190-90074), which describes the HP changes and additions to OSF 1.2.1; it also describes installing and configuring HP DCE 1.8 and how to migrate from previous releases of HP DCE to HP DCE 1.8. This document describes both HP DCE 1.8 clients and servers.

All other documentation for these products are available online only; you can print the online files through the standard help and man page print functions.

Online Documentation

DCE-C-Tools

DCE-C-Tools includes the following online documentation:

- The following CDE Help volumes:
 - ✓ *HP DCE Application Development Tools* (access to man pages)
 - ✓ *HP Sample Applications Online Help*To access these volumes, select *HP DCE Application Development Tools* from the CDE Front Panel.
- *HP DCE Version 1.8 Application Development Tools Release Note*, an ASCII version of this document can be found in the following path: `/opt/dce/newconfig/Relnotes`.
- *Man pages*

NOTE To access these man pages from the HP Help System, include the directory `/usr/share/man` in your MANPATH environment variable.

NOTE Refer also to the ASCII README file included in the directory of each of the sample applications for important information about that application.

Known Problems and Workarounds

This section describes known problems and workarounds in the HP DCE version 1.8 Application Development Tools for HP-UX 11i.

- **SR - No 8606-165933**
Secd does not log trace for sec component via dcecp command
- **SR - No 8606-169328**
cdsd returns wrong ACL to dcecp

DCE-Core Tools

As of DCE 1.6, the **DCE_SVC_DEBUG** macro was changed to acquire a SVC mutex lock (in prior releases this was done in the serviceability library). Because of this, DCE applications based on DCE 1.5 and earlier releases that make use of this macro will be binary incompatible with DCE 1.6 applications and so must be recompiled under DCE 1.6 on HP-UX 10.30.

DCE-C-Tools

There are no problems in HP DCE-C-Tools discovered at HP DCE 1.5, 1.7, or 1.8.

Compatibility Information and Installation Requirements

The section provides a brief overview of the installation process for the HP DCE Version 1.8 Application Development Tools for HP-UX 11i and the software prerequisites and disk space requirements that must be met before you can install the software.

Installation Requirements

The following is a brief overview of the installation process:

1. Verify that your site meets the hardware and software prerequisites described in the following subsections.
2. Use the swcopy tool to load HP DCE/9000 (Version 1.8) Application Development Tools, CDS Server or Security Server software from the media to the same network distribution area that contains the HP DCE/9000 Core Services release software.
3. Use the swinstall tool to install the required products on individual systems.

Depending on which license(s) you have purchased from Hewlett-Packard, you can install the respective filesets from distribution media.

Prerequisites

The prerequisites for installation of DCE Version 1.8 Application Development Tools for HP-UX 11i are:

DCE-C-Tools

- HP Series 700/800 Computer
- HP-UX 11i release, including the following filesets:
 - ✓ DCE-Core.DCE-CORE-RUN
 - ✓ DCE-Core.DCE-CORE-SHLIB
 - ✓ DCE-CoreTools.DCE-BPRG

The first two filesets listed are included with the HP-UX 11i Core OS media. The last fileset is included with the HP-UX 11i Application Release media. (At HP-UX 10.20 and previous releases, it was part of the Core OS.)

HP DCE/9000 Version 1.8 Application Development Tools for HP-UX 11i Release Note
Compatibility Information and Installation Requirements

- HP C Compiler (Version A.11.0)
- The DCE-C-Tools Programming Libraries filesset, DCE-TOOLS-LIB

NOTE The HP DDE debugger is highly recommended for multithreaded debugging support.

Patches and Fixes in This Version

The following fixes are included in this version:

Table 1-1. Fixes

SR Number	Known Problem
SR-No 8606-114757	The <i>pthread_setasynccancel</i> man page needs to be updated to mention cancellation points.
SR - No 8606-126271	In certain header files the thread <i>_POSIX_C_SOURCE</i> had been incorrectly set to " <i>POSIX_D10_THREADS</i> "
SR - No 5003-264093	Some cds operation makes multiple read-only replicas of a directory
SR - No 8606-113871	cdsbrowser dumps core after <i>XtFree()</i> call
SR - No 8606-129648	secd dumped core file in <i>free()</i> because the header information got corrupted
SR - No 8606-110658	On the N Class machine with 9 LAN interfaces the cdsd dumps core
SR - No 8606-155974	Can't slulk any more once 'a' permission is lost

Patches

All the fixes that have been integrated in the DCE 11.0 patches have been included in the HP DCE 1.8 release. Refer to the HP patch catalog for more details:

- PHSS_20447
- PHSS_17813

Features to be Obsoleted in the future releases of DCE

The list of DCE features/tools/libraries that are to be obsoleted are given below:

NOTE The following obsoleted DCE features/tools/libraries will not be supported in the future releases.

- /bin/sec_clientd
- /bin/cdsclerk
- /bin/nidl_to_idl
- /bin/idl++
- /hptools/bin/i2dl
- /bin/dcm/ **and related deliverables**
- /lib/libbb.a
- DMS feature
- libsis.sl
- account manager
- libdce.1
- All the archived libraries. (libd4r.a, libdcekt.a, libcma.a, libdce.a)

Software Availability in Native Languages

The HP DCE 1.8 Application Development Tools for HP-UX 11i are currently available in English only.

Developing DCE Applications with HP DCE/9000

This chapter provides information about developing, building, and debugging DCE applications with HP DCE/9000.

NOTE The HP DCE application development software, including the IDL compiler, the NIDL-to-IDL translator, DCE interface definition files and header files, the International version of the DCE archive library, and the pthreads archive library, may not be included in the HP-UX bundle that you installed. If you intend to develop HP DCE applications, you must explicitly install any software that you require.

Building DCE Programs

NOTE Hewlett-Packard supports only the ANSI C compiler for building HP DCE applications. Hewlett-Packard cannot provide support for problems with HP DCE applications not compiled using ANSI C. This restriction also applies to applications on HP-UX 10.x systems built using the HP-UX user-space threads library (**libcma**).

When compiling and linking HP DCE applications, note the following:

- In order for the correct header file contents to be used, define **_HPUX_SOURCE** when compiling your HP DCE application: **-D_HPUX_SOURCE**.
- In order for the correct header file contents to be used, define **_REENTRANT** and **_PTHREADS_DRAFT4** when compiling HP DCE programs as **-D_REENTRANT** and **-D_PTHREADS_DRAFT4** for the following reasons:
 - ✓ To ensure that the threadsafe routines such as **putc**, **putchar**, **getc**, and **get char** are used instead of the non-thread-safe macro versions defined in **stdio.h**.
 - ✓ To get definitions of new structures and to provide ANSI C prototype information for the new reentrant interfaces.
- If you include **<pthread.h>**, you must do so before other header files in your C source file. Also note that **<pthread.h>** defines **_REENTRANT**.

- Source code that is built into applications that use the CDS, RPC, or security APIs must include `<pthread.h>`. This is necessary because the DCE RPC runtime library creates a small number of private threads, on both the client and server sides of an application.
- The DCE header files should be included as `<dce/ header.h>`. Because names of some of the DCE header files conflict with those in `-Iusr/include`, you should avoid using the `-Iusr/include/dce` C preprocessor option to include DCE header files.
- DCE applications must always link with **libdce**. When linked with other libraries, **libdce** must be linked first, as follows: `-ldce other_libraries`

DCE applications that require kernel thread support needed to link to **libcektc** as follows: `-ldcekt`

DCE applications requiring 64 bit, kernel thread support need to link to `libdcekt` after compiling the binaries using the guidelines specified in "<http://www.software.hp.com/STK>". This site contains the 64 bit porting guidelines, 64 bit compiler and linker changes needed to port the applications of 64 bit.

Do not link in **libc** explicitly when building a DCE application.

- If you link in **libdce**, **libcma** is automatically linked in; if you are using **libcma** without **libdce**, you must link **libcma** first, as follows: `-lcma other_libraries`
- **libdce** and **libc** must be either both shared or both archived library versions; a mixed environment with one of the libraries the shared version and the other the archived version is not supported.
- **libcma** and **libc** must be either both shared or both archived library versions; a mixed environment with one of the libraries the shared version and the other the archived version is not supported.

NOTE At HP DCE 1.8, both **libdce** and **libcma** were versioned for compatibility reasons. **libdce.1** and **libcma.1** are the latest patched HP DCE 1.5 libraries. **libdce.2** and **libcma.2** support HP DCE 1.8 on HP-UX 11i.

 Shared applications built on HP DCE 1.7 may have to recompile to run on HP DCE 1.8.

Notes on Debugging HP DCE Applications

The following are tips on debugging HP DCE applications. Also see *Programming with Threads on HP-UX (B2355-90060)* for additional information about debugging HP DCE

Applications.

- To debug an HP DCE threads application with DDE, use the **-dce_thread** option. By default, DDE assumes that the application is single- or kernel-threaded.

The **debug** and **dde** commands have been extended to take the **-dce_thread** option. For example:
`debug -dce_thread averagedde -dce_thread average`

See Section 3.2.1 (DDE) of the *HP-UX Programming Tools Release Notes (5965-4409)* and the *dde* man page for more information about the **-dce_thread** option.

- HP DCE uses the HP-UX SIGCHLD and SIGVTALRM signals internally. When debugging HP DCE servers or clients, you must ensure that these signals, when received, are passed along to the target process. If you are using DDE or **xdb**, you can modify your debugger start-up files as follows:

```
~/.dderc: alias `after_debug [ del int signal SIGCHLD;del int signal SIGVTALRM]
```

```
~/.xdbrc: z 18 rs # do not stop or report SIGCHLD z 20 rs # do not stop or report SIGVTALRM
```

See *Programming with Threads on HP-UX* and the discussion of **-signal()** in this chapter for more information on the use of signals with HP DCE applications.

- Be sure the application has an exception handler installed that can field any RPC exceptions.
- Run the server side of your application under the debugger so that exceptions raised in server code will trap into the debugger rather than being reflected back to the client process via RPC. This makes it easier to identify bugs in the server that might otherwise appear to be client bugs.

See the README file and **server_debug.ksh** script in

`/opt/dce/share/hpexamples/string_conv` for information on debugging "dced-started" servers via **xdb** or **dde**.

Using HP DCE with C++ Applications

If you want to use C++ with standard DCE to create DCE applications, there are some practices you must avoid. Most of these relate to lack of thread-safeness of C++ operations.

The following list describes the practices you should avoid. These guidelines apply to HP C++ compiler version A.3.70 and later.

By taking these precautions, you should be able to productively use C++ to write DCE applications.

- When using unsafe constructs, be sure to use the constructs in only one thread at a time.

- DCE CMA exceptions should never be allowed to propagate into a C++ scope. Allowing this to happen could result in destructors failing to be executed. This can lead to memory leaks and unexpected behavior. For example, suppose some C++ code makes a call to a C function. Within this C function, a CMA exception can be raised. It is very important that the CMA exception also be caught within the C function. If it is not, then the CMA exception could propagate into the C++ environment, resulting in unexpected behavior.

A DCE exception can also be caught in a C++ routine which has called a DCE function. You must be sure that there are no memory allocations within the TRY block (note that some conversions cause memory allocation):

```
Try {call a function which might do a DCE RAISE}  
CATCH_ALL {handle it but don't RERAISE}  
ENDTRY
```

Note that some library calls, and most system calls, are CMA cancellation points, and must be wrapped as described here.

- Do not enable cancellation of a thread that is running C++ code. Asynchronous cancellation may cause a DCE exception to be raised at any time. Synchronous cancellation may cause a DCE exception to be raised any time a system routine is executed.
- The following item applies only when programming DCE applications using C++ and standard DCE:

Do not use C++ exceptions in multi-threaded processes, including DCE servers and managers. The stack unwinding that occurs when a C++ exception is raised is not guaranteed to work in the presence of multiple RPC threads.

Notes on Programming with HP DCE

The following are miscellaneous notes on programming with HP DCE. Also see *Programming with Threads on HP-UX* (B2355-90060).

Several features of interest to programmers are not supported at the HP DCE 1.8, including support for the **rdacif** routines for ACL managers. You should also be aware of future changes to threads support. For details, see "Features Planned for a Future Release" in Chapter 1 of *Planning and Configuring HP DCE 1.8*.

Appropriate Uses of the Cell Directory Service

The Cell Directory Service (CDS) was designed for a specific purpose: to store and retrieve server bindings in a DCE cell. This mission implies numerous assumptions about the volume of information that must be stored and about the types and frequency of access to that

information. Performance trade-offs in CDS's usage of memory, disk, cache, the network, and time-outs are optimized to this very specialized purpose.

The generality of the application programming interfaces to CDS masks the specificity of this purpose, and gives CDS the appearance of a general-purpose database system. It is not; the optimizations of CDS for its low-volume and weakly-replicated storage, and for retrieval of server bindings in a fairly static name space, are rarely optimal for other purposes. Moreover, other uses of CDS directly compete and interfere with its use and availability as a critical core component of the DCE system.

Application developers are frequently tempted to use CDS for the direct storage of information other than server bindings. For example, the developer of a telephone number directory service may consider using CDS to store the information directly. This is a bad idea, as the volume of information could overwhelm the in-memory CDS data structures, and very frequent read accesses could slow CDS performance and lock out DCE server lookup requests. A much better strategy would be to implement a database server designed specifically for the telephone number directory service, and then store the bindings of that server in CDS for lookup by clients. In this way, other uses do not compromise CDS in its intended use as a critical DCE component.

Cell Directory Service Programming Interfaces

The supported programming interfaces to CDS are the RPC Name Service Independent (RPC_NS) interface and the X/Open Directory Service (XDS) interface. The RPC_NS function call prototypes are in the include file `/usr/include/dce/rpc.h`. The XDS function call prototypes are in the include file `/usr/include/xds.h`. The RPC_NS interface is the procedural interface used by the core DCE components and most applications. The XDS interface is an object-oriented interface appropriate for use with other X/Open standards such as X.500. Both the RPC_NS and XDS interfaces support many common name space functions, but lack administrative capabilities such as the ability to create and delete directories and manage replication.

dcecp is the recommended tool for performing CDS administrative functions such as creating and deleting directories and managing replication. **dcecp** can be called in a shell script, but is not a programming interface.

CDS_DELAY_REPLICA_CREATE

When creating the directory and its readonly replica continuously (i.e., without any time gap), the creation of replica is started before the creation of Master is completed. So an error "requested entry does not exist" occurs. To avoid this export the environment variable `CDS_DELAY_REPLICA_CREATE` which will delay the creation of the replica.

Usage of this environment variable:-

Run the command "export `CDS_DELAY_REPLICA_CREATE=sometimestamp`". Here the

timestamp should be between 0 to 5000 ms (milliseconds). Typically 2000 ms is used in most cases.

RPC_RESTRICTED_PORTS Environment Variable

The capability to restrict the assignment of endpoints to those in a user-specified set was added to RPC in OSF DCE 1.0.3. This allows DCE applications to operate in environments in which inter-network traffic is restricted to specified endpoints. The facility is activated by setting the `RPC_RESTRICTED_PORTS` environment variable with the list of end points to which dynamic assignment should be restricted before starting an RPC application. `RPC_RESTRICTED_PORTS` governs only the dynamic assignment of server and client ports by the RPC runtime. It does not affect well-known endpoints.

The facility is turned on by setting the `RPC_RESTRICTED_PORTS` environment variable before starting an RPC application. The syntax of the variable is as follows:

```
<entry> [COLON <entry>]*  
<entry> : <protseq_name>  
LEFT-BRACKET <ranges> RIGHT-BRACKET  
<ranges>: <range> [COMMA <range>]  
*<range> : <endpoint-low> HYPHEN <endpoint-high>
```

For example:

```
ncacn_ip_tcp[5000-5110,5500-5521]:ncadg_ip_udp[6500-7000]
```

To use `RPC_RESTRICTED_PORTS` for DCE itself, set the environment variable before starting your cell. The environment variable must be set whenever you restart DCE.

Note that this facility does not add any security to RPC and is not intended as a security feature. It merely facilitates configuring a network "fire wall" to allow incoming calls to DCE servers.

RPC Authentication

The *OSF DCE Application Development Guide* and the *OSF DCE Application Development Reference* may be misleading about what happens when an unauthenticated client calls a server that has specified authentication. In such a case, the RPC runtime will not perform any authentication, and the call will either reach the server manager code, or be rejected by the runtime, depending on the following conditions:

- If the client specified no authentication, then none is attempted by the RPC runtime. The call reaches the manager code whether the server specified authentication or not. This permits both authenticated and unauthenticated clients to call authenticated servers. When the manager receives an unauthenticated call, it must make a decision about how to proceed.

- If the client specified DCE secret key authentication and the server specified no authentication, then the runtime will reject the call, and it will never reach the manager routine.
- If both client and server specified DCE secret key authentication, then authentication will be carried out by the RPC runtime transparently. Whether the call reaches the server manager code or is rejected by the runtime will depend on whether the authentication succeeds.

Although the RPC runtime is responsible for any authentication that is carried out, the fact that the runtime will always permit unauthenticated clients to reach the manager code means that a manager access function typically does need to make an authentication check. When the manager access routine calls `rpc_binding_inq_auth_client()`, it should check for a return status of `rpc_s_binding_has_no_auth`. When such a status is returned, it means that the client has specified no authentication, and the manager access function will have to make an access decision based on this fact. Note that in such a case, no meaningful authentication or authorization information is returned from `rpc_binding_inq_auth_client()`.

RPC Data Transfer Limitation

The bulk data transfer (for example, IN/OUT-pipes) over the connection-oriented (TCP/IP) RPC protocol is limited by the performance difference between the client and server machines. If the receiver process is significantly slower than the sender process (and cannot process data fast enough), the receiver process's virtual memory usage may grow rapidly until the remote procedure call fails with an `rpc_s_no_memory` status.

Restricting RPC Addresses

The runtime looks for a `RPC_SUPPORTED_NETADDRS` environment variable, which allows a user or administrator to restrict the network addresses that a DCE server will advertise in the name space/endpoint-map.

If this environment variable is set, only addresses in the list will be advertised in the name space or endpoint map. Addresses not found on the list will be excluded from the server's list of available addresses.

The format of the `RPC_SUPPORTED_NETADDRS` string is as follows:

```
RPC_SUPPORTED_NETADDRS=PROTSEQ:netaddr[ ,protseq:netaddr ]
```

For example, assuming that host **myhost** is located at IP address 10.3.2.1, the Korn shell statements:

```
export RPC_SUPPORTED_NETADDRS=ip:myhost
```

or

```
export RPC_SUPPORTED_NETADDRS=ip:10.3.2.1
```

will force any servers started in the current shell to support only the addresses associated with the name **myhost** and the network address 10.3.2.1.

Calling `exec()` from a DCE Application

Care must be used when calling `exec()` from a DCE application. HP DCE Threads sometimes sets open file descriptors to non-blocking mode, so that I/O calls block only the calling thread, not the entire process. This occurs unbeknownst to the application itself. HP provides wrappers for the `exec()` family of calls that, among other things, resets file descriptors to blocking mode if they were set non-blocking by HP DCE Threads and not the application.

For file descriptors that were inherited across `fork()`, this also has the effect of resetting the file descriptor to blocking mode in the parent as well as the child. If the parent is a threaded program, it can cause the parent to hang due to a blocking I/O call.

There are two possible work-arounds:

- If the process will not need the open file descriptor, set the close-on-exec flag for the file prior to calling `exec()`. The file descriptor will not be reset in this case.
- Avoid using the `exec()` wrapper; call `exec()` directly instead. This is accomplished by undefining the appropriate macro, for example: `#undef execl`.

Note that if the `exec()` wrapper is circumvented, the new process may inherit file descriptors that are unexpectedly set non-blocking; some signals may be unexpectedly ignored or not ignored; and, if `exec()` is called without first calling `fork()`, the new process will probably be killed by SIGVTALRM as soon as it begins execution.

Process Forking

Process forking from within RPC applications that use the connection-oriented (TCP/IP) RPC protocol is not supported.

While it is generally safe for an application to perform a fork followed immediately by an `exec()`, the following sequence may not work for TCP/IP RPC programs:

- The TCP/IP RPC process forks.
- The child process tries to use RPC over the TCP/IP protocol before the `exec()`.

Process forking from within RPC applications that use the connectionless protocol (UDP/IP) is supported, with the following restrictions:

- For client-side applications, the UDP/IP protocol is fork safe. It is the responsibility of the application developer to ensure that all other application threads are capable of crossing forks safely.

- On the server side, the only supported behavior is for a server thread to fork and exec, with no use of RPC in the child of the fork until after the exec.

File Locking

The HP DCE Threads **fcntl()** wrapper does not provide for thread-synchronous file locking. The entire process will block when a call to **fcntl()** specifies **F_SETLKW** and the file is currently locked by another process.

Note also that HP-UX file locks are a process-wide resource. For this reason, if multiple threads call **fcntl()** to lock the same section of a file, all the calls will succeed, and each thread will believe it holds the lock. However, only a single, process-wide lock is actually obtained — the first call obtains the lock, the second and subsequent calls have no effect. If one thread releases this lock, the other threads will continue to believe the file is locked, when in fact the process no longer holds any lock on that section of the file.

Use mutex locks to ensure that only one thread at a time locks a particular file or section of a file.

The previous comments regarding file locking with **fcntl()** also apply to **lockf()**. **lockf()** is not wrapped by DCE Threads.

sec_id_parse_name()

CAUTION In previous HP DCE releases, the **sec_id_parse_name()** library call could be passed a NULL pointer or null string for the global principal name and would return the local cell name. This behavior was maintained in HP DCE 1.6, but is not supported in HP DCE 1.7. Instead, pass **"/.:"** to obtain this result.

sec_login_valid_and_cert_ident()

The **sec_login_valid_and_cert_ident()** routine uses **fcntl()** for file locking, and should not be called by more than one thread at a time. If you must use **sec_login_valid_and_cert_ident()** in multiple threads, use a mutex to insure that only one thread at a time executes the call. No HP DCE programming interfaces call **sec_login_valid_and_cert_ident()** internally.

Applications that need to obtain credentials in multiple threads will generally only need to call **sec_login_validate_identity()**, which is not affected by **fcntl()**.

semop()

The HP DCE Threads **semop()** will not increment the *semncnt* or *semzcnt* kernel variables.

Because the **semop()** wrapper adds the `IPC_NOWAIT` option before performing the **semop()** system call, the value of `semncnt` and `semzcnt` should not be trusted when using **semop()** to perform semaphore operations.

signal()

Use of the **signal()** system call is not supported by HP DCE Threads, as it can interfere with signal handlers that are installed by Threads. In some cases, you can retain **signal()** calls in legacy code as follows:

- Implement a wrapper for **signal()** that calls **sigaction()** or - **sigwait()**.
- Use **signal()** itself to install a handler for a signal, provided the signal is not `SIGVTALRM`, `SIGCHLD`, or `SIGSYS`, and provided no thread installs a handler for the signal with **sigaction()** or waits for the signal with **sigwait()**.

system()

The HP DCE Threads **system()** wrapper does not block `SIGCHLD`. It does set the handler for `SIGQUIT` and `SIGINT` to `SIG_IGN`, but only for the calling thread. This behavior differs from the behavior of the standard HP-UX version of **system()**.

vfork()

The HP DCE Threads **vfork()** wrapper cannot be circumvented. The methods used to circumvent other wrappers (defining `_CMA_NOWRAPPERS_`, undefining **vfork**, defining **vfork** to be `_vfork_sys`) do not work.