IBM Distributed Computing Environment Version 3.1 for Solaris:

# Quick Beginnings

IBM Distributed Computing Environment Version 3.1 for Solaris:

# Quick Beginnings

**Note**
Before using this document, read the general information under “Appendix C. Notices” on page 91.

**First Edition (August 1999)**

This edition applies to Version 3.1 of *IBM Distributed Computing Environment for Solaris* and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

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

IBM welcomes your comments. Send your comments to the following address:

International Business Machines Corporation
Department VLXA
11400 Burnet Road
Austin, Texas
78758

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

# Contents

# Tables



 

# Welcome to DCE 3.1 for Solaris

This book describes the IBM® Distributed Computing Environment for Solaris, Version 3.1 (DCE 3.1 for Solaris), and explains how to plan for, install, and configure the product.

"Part 1. Understanding DCE 3.1 for Solaris" on page 1 gives an overview of DCE 3.1 for Solaris.

"Part 2. Planning for and Installing DCE 3.1 for Solaris" on page 15 provides planning, installing, and configuring information. This book provides information for server components and client components.

"Part 3. Configuring, Starting, and Stopping DCE 3.1 for Solaris" on page 53 explains how to use DCE 3.1 for Solaris.

## Typographic and Keying Conventions

This guide uses the following typographic conventions:

**Bold** **Bold** words or characters represent system elements that you must use literally, such as commands, options, and pathnames.

*Italic* *Italic* words or characters represent variable values that you must supply. *Italic* type also introduces a new DCE term.

`Constant width`
Examples and information that the system displays appear in `constant width` typeface.

**[ ]** Brackets enclose optional items in syntax descriptions and format.

**{ }** Braces enclose a list from which you must choose an item in syntax descriptions and format.

**|** A vertical bar separates items in a list of choices.

**< >** Angle brackets enclose the name of a key on the keyboard.

**...** Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.

This guide uses the following keying conventions:

**<Ctrl-** *x***> or ^** *x*
The notation **<Ctrl-** *x***>** or **^** *x* followed by the name of a key indicates

 

a control character sequence. For example, **<Ctrl-C>** means that you hold down the control key while pressing **<C>**.

**<Return>**

The notation **<Return>** refers to the key on your terminal or workstation that is labeled with the word Return or Enter, or with a left arrow.

# Part 1. Understanding DCE 3.1 for Solaris

# Chapter 1. Overview of DCE 3.1 for Solaris

IBM Distributed Computing Environment for Solaris, Version 3.1 (DCE 3.1 for Solaris) is a member of the IBM Server Series family of products. The DCE 3.1 for Solaris is based on OSF Distributed Computing Environment (DCE) technology (Release 1.2.2).

## What Is DCE?

DCE provides a standard environment that supports distributed applications. It represents technologies that are selected by the OSF and has emerged as the leading industry standard for distributed services.

An application written to use DCE runs in any environment that supports the OSF DCE standard. DCE makes it possible for application developers to give users secure access to the wide range of information and services available within their network and also hides the complexity of the network environment.

Distributed computing services, as implemented in DCE, provide an important enabling software technology for the development of distributed applications. DCE makes the underlying network architecture transparent to application developers. It consists of a software layer between the distributed application program and the operating system and network interface. DCE provides a variety of common services that are needed for development of distributed applications, such as name and time services, and a standard remote procedure call (RPC) interface. DCE provides a means for application developers to design, develop, and deploy distributed applications.

A group of DCE machines that work together and are administered as a unit is called a *cell*. For example, imagine an organization comprised of several departments, each in a different building and operating on its own budget. Each department in such an organization could have its own DCE cell.

A DCE environment is a group of one or more DCE cells that can communicate with each other. A cell becomes a part of a DCE environment when it obtains access to one or more global directory services in which the other cells in the environment are registered.

If two cells for two different departments are a part of a DCE environment, then a user in one department's cell can access resources in another

department's cell. This access is typically less frequent and more restricted than access to resources within the user's own cell.

You can configure a DCE cell in many ways, depending on its users' requirements. A cell consists of a network that connects two kinds of nodes:

- **DCE user (client) machines** are general-purpose DCE machines. They contain software that enables them to act as clients to all of the DCE services.
- **DCE server machines** have special software that enables them to provide one or more of the DCE services. Every cell must have at least one of each of the following servers in order to function:
  - Cell Directory Server
  - security server

Other DCE servers can be present in a given DCE cell to provide additional functionality. For example, a Global Directory Agent can enable the cell's directory server to communicate with other cells' directory servers.

DCE 3.1 for Solaris is a layer between the Solaris operating system, network services, and a distributed application. It provides the services that allow a distributed application to interact with a collection of possibly heterogeneous computers, operating systems, and networks as if they were a single system. DCE 3.1 for Solaris includes a set of standard services, software interfaces, and tools that support the creation, use, and maintenance of distributed applications in a diverse computing environment.

DCE 3.1 for Solaris has the same organization as OSF DCE. Part 1 of this book introduces the concept of a DCE cell and gives a brief summary of how a Distributed Computing Environment organizes participating machines.

DCE 3.1 for Solaris is based on the OSF DCE Release 1.2.2 code base and designed for the supported versions of the Solaris operating system. See the *IBM DCE Version 3.1 for Solaris: Release Notes* for a listing of the supported versions of the Solaris operating system.

## Product Contents

DCE 3.1 is available in the following packages:

- **DCE for Solaris, Version 3.1** (DCE 3.1 for Solaris) which includes the following:
  - **DCE Base Services for Solaris, Version 3.1**
  - **DCE Security Services for Solaris, Version 3.1**
  - **DCE Cell Directory Services for Solaris, Version 3.1**

- **DCE Base Services for Solaris, Version 3.1** (DCE 3.1 Base Services)
- **DCE Data Encryption Standard Library for Solaris, Version 3.1** (DCE 3.1 DES Library) which includes:
  - **DCE Base Services for Solaris, Version 3.1**

## DCE 3.1 Packages

### DCE 3.1 for Solaris

**DCE 3.1 for Solaris** consists of the following packages:

- **DCE Base Services for Solaris** provides support for remote procedure calls, the client functionality for cell directory service and security, time, messaging and serviceability. This package also provides support for integrating DCE security services with Solaris base operating system security. DCE includes administration tools for such functions as configuring a cell, adding and deleting cell users, and adding servers and clients to a cell.
  - **Client Services**
    - The **Remote Procedure Call (RPC)** facility enables you to create and run client applications and server applications. The RPC runtime service implements the network protocols by which the client and server sides of an application communicate.
    - **DCE Threads Compatibility Library for Solaris** provides a programming model for building concurrent applications that perform many operations simultaneously. It provides support for multithreaded applications (based on POSIX 1003.4a Draft 4) that use the DCE threading model. The Solaris package includes a DCE Threads Compatibility Library for Solaris.
    - **Multithreaded Programming Environment** support allows multiple threads to call standard C library functions without interfering with one another.
    - **Distributed Time Service (DTS)** provides synchronized time in the distributed network environment on the computers that participate in a Distributed Computing Environment. DTS synchronizes a DCE host's time with Coordinated Universal Time (UTC), an international time standard.
    - The **CDS client** provides the interface, **cdsclerk**, between CDS client applications and CDS servers. The **slim client** also provides the **cdsclerk** interface and most of the functionality of the regular client, but because **dced** is not executing on slim client machine, no endpoint processing can occur.
    - The **security client** provides the following services:

- **Solaris Security Integration** coordinates the Solaris base operating system security services with the DCE Security services. This consists of the Pluggable Authentication Module (PAM) and the Name Service Switch (NSS). Solaris Security Integration allows a user to log in to Solaris and obtain DCE credentials at the same time. For more information about Solaris Security Integration, see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components*.
- **GSSAPI Extensions** — GSSAPI extensions are a set of APIs that provide non-RPC applications the ability to use the DCE security authentication protocol. The GSSAPI extensions can be used to establish credentials or extract Extended Privilege Attribute Certificates (EPAC) for a non-RPC application.
- **Extended Registry Attributes** (ERA) — This expands the static registry attributes of principal, group, and account to a dynamic set of registry attributes that can be customized to a cell.
- The **Audit Service** logs audit records based on specified criteria. The Audit Service has three basic components:
  - **Application Programming Interfaces** provide the functions that are used to detect and record critical events when the server services a client. They also are used to create tools that examine and analyze the audit event records.
  - **Audit Daemon** maintains the filters and the audit logs.
  - **Audit Management Interfaces** are used by the Administrator to specify how the Audit Daemon will filter the recording of Audit Events. This interface is available from the DCE Control Program (**dcecp**).
- The **Enhanced Password Strength Server** extends the capabilities of the password strength server in previous DCE releases. The enhanced server allows you to control the following characteristics of user passwords:
  - Password composition
  - Password age
  - Password history and re-use
  - Password dictionaries and user-defined rules

- **CDS Preferencing** enables administrators to specify a preferenced CDS clearinghouse from which a client will obtain CDS information. This feature is provided to improve performance at CDS clients by enabling cell administrators the ability to specify a preferred CDS clearinghouse from which a client will obtain CDS information. This is useful in situations where, for example, a low-performance WAN connects multiple high-performance LANs, and each of the LANs contains a CDS replica clearinghouse. With this feature, administrators can specify local

clearinghouses as preferred over distant clearinghouses, and then clients use the distant clearinghouses only when the local clearinghouses are unable to satisfy a request.

– **DCE Web Secure** provides DCE credentials to CGI programs. DCE Web Secure must be installed and configured on a workstation that has a Netscape Enterprise 3.61 or a FastTrack 3.01 Web server and a DCE client within the cell.

– The **Online Documentation for DCE 3.1 for Solaris** was enhanced to provide the following:
  - An online IBM documentation set in HTML format
  - An IBM documentation set in PDF format

  For more information concerning the documentation, refer to "Chapter 6. Obtaining Additional Information" on page 79.

– **DCE System Management** provides two management tools: DCE Event Management Services and the DCE SNMP SubAgent.
  - **Simple Network Management Protocol (SNMP)** provides network management support in the TCP/IP environment for monitoring DCE resources and services. System administrators and system management application programmers can use SNMP to easily monitor the DCE environment so that they can focus on making their resources and services more manageable. For more information about SNMP, see the *IBM DCE Version 3.1 for Solaris: Application Development Guide—Introduction and Style Guide*.
  - **Event Management Service (EMS)** provides asynchronous event support for DCE based applications. DCE EMS manages event services in a DCE cell. EMS consists of two parts — the **emsd** (EMS daemon) server and APIs to access event services through an interface to the suppliers, consumers, and event service administration for use by EMS clients. For more information about EMS, see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components*.

– **DCE for Application Developers (dcetools)** includes tools for DCE administrative and application development support. The Interface Definition Language tool consists of a language (and its compiler) that supports the development of distributed applications that follow the client and server model. It automatically generates code that transforms procedure calls into network messages. The **sams** compiler generates files that can be used to add messaging and serviceability support to DCE applications.

– The **DCE XDS/XOM for Solaris** provides application programming interfaces to the CDS namespace. A library of functions is available with which to access the Directory Services.

- **DCE Security Services for Solaris** enables secure communications and controlled access to resources. It provides a set of the following security-related functions:
  - **Authentication Service** — enables two processes on different machines to be certain of each other's identity.
  - **Secure Communication** in which communication is protected by the integration of DCE RPC with the security service.
  - **Authorization** in which access to resources is controlled by comparing the credentials conferred to a user by the Privilege Service with the rights to the resource, which are specified in the resource's Access Control List (ACL).
  - **Privilege Server** — once identity has been established, the following checks are made: Is the user authorized to access a resource? What permissions are required, and does the user have those permissions? Authentication and authorization are generally invoked for the user through use of Authenticated RPC.
  - **Access Control List Facility** — ACLs are lists of users who are authorized to access a given resource. An ACL API allows programmers to manipulate ACLs, and the dcecp **acl** commands or the **acl_edit** command allow users to modify ACLs associated with resources that they own, to whom (user or group) access is granted and what specific permissions are given.
  - **Login Facility** — initializes a user's DCE security environment by authenticating the user to the security service by means of the user's password, then by returning security credentials that will authenticate the user to the required distributed services.
  - **Public Key certificate login** allows a user to obtain initial DCE credentials by using an X.509v3 digital certificate and its associated public key pair to prove the user's identity. This feature is an extension of the OSF DCE 1.2.2 public key login protocol based on OSF RFC 68.4 (draft 7). This implementation requires the Entrust public key infrastructure (PKI)
  - **Security Replication** — enables the master Registry Database to be replicated to one or more subordinate Registry Databases. The dcecp **registry** commands or the **sec_admin** command are the interfaces used to view and manipulate the state of both master and subordinate replicas.
  - **Identity Mapping Service** — is used by the Authentication Service to determine a DCE user's identity when the user logs in with the public key certificate authentication protocol. The identity mapping service maps a user's name in a public key certificate to a DCE principal name.
- The **Directory Service for Solaris** is a central repository for information about resources in the distributed system. Typical resources are users,

machines, and RPC-based services. The information consists of the name of the resource and its associated attributes. Typical attributes include a user's home directory or the location of an RPC-based server.

The Directory Service consists of the Cell Directory Service (CDS) and the Global Directory Agent (GDA). The Cell Directory Service manages a database of information about the resources in a group of machines called a DCE cell and provides location-independent naming for servers. The GDA enables intercell communications by locating cells which have been registered in the global naming environment.

**GDA Integration with LDAP** is an extension to GDA that allows the resolution of non-DNS style foreign cell names. X.500 directories and any directories that support the LDAP protocol can be used to establish intercell communication. For more information about LDAP, see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components.*

### Data Encryption Standard

**Data Encryption Standard** (the **IDCEpriv** feature) provides a programming interface that enables remote procedure call (RPC) application data encryption. The **IDCEpriv** feature utilizes the Data Encryption Standard (DES) algorithms that are part of the DCE Base Services for Solaris. This feature includes support for User Data Masking Encryption, which was formerly packaged separately.

## IBM Enhancements to DCE

The following services and commands contained in the previously listed DCE 3.1 packages are IBM enhancements and extensions to the Solaris implementation of DCE:

- **Added Services**:
  - **User Data Masking Encryption Facility**
  - **Enhanced Password Strength Server**
  - **CDS Preferencing**
  - **DCE Web Secure**
  - **Simple Network Management Protocol (SNMP)**
  - **Event Management Service (EMS)**
  - **Solaris Security Integration**
  - **Public Key Certificate Login**
  - **GDA Integration with LDAP**
  - **Identity Mapping Server (IDMS)**
  - **Slim Client**
- **Additional Commands**:

– Configuration Commands:

**Notes:**

1. Use the new command format, however, the old command format is still supported.
2. These commands are not compatible with the dcecp **host configure** and **host unconfigure** commands.

**chpesite**
: Updates the **pe_site** file, which contains the addresses of the security servers that you use.

**clean_up.dce**
: Cleans up recreatable database files, cache files, and credential files. This command is intended to be used if problems are encountered when trying to start DCE.

**config.dce**
: Configures and starts DCE components. This command provides for a split configuration of clients. Administrative configuration and local configuration can be performed separately. See "Further Cell Configuration" on page 65 for more information.

**kerberos.dce**
: Creates the host principals, FTP principals, and key table entries that are used to support kerberos interoperability with DCE.

**mkdceweb**
: Configures DCE Web Secure into a Netscape FastTrack 3.01 or Enterprise 3.61 Web server.

**mkreg.dce**
: Adds information about a DCE cell into the DOMAIN namespace.

**rmdceweb**
: Unconfigures DCE Web Secure from a Netscape FastTrack 3.01 or Enterprise 3.61 Web server.

**rmreg.dce**
: Removes information about a DCE cell from the DOMAIN namespace.

**show.cfg**
: Displays the local host's DCE or DFS configuration or both configurations. The **dce** and **dfs** options allow display of only DCE or DFS information

**start.dce**
: Starts the configured DCE components. This command makes sure that all components are started in the correct order.

**stop.dce**
: Stops the configured DCE components. This command makes sure that all components are stopped in the correct order.

**unconfig.dce**
: Removes configuration of DCE components. This command

provides for a split unconfiguration, with which administrative unconfiguration and local unconfiguration can be performed separately. See “Further Cell Configuration” on page 65 for more information.

– Cell Directory Service (CDS) Commands:

**catraverse**
Traverses the clerk cache.

**cds_dbdump**
Dumps CDS server database.

**cdsd_diag**
Starts the CDS Diagnostic utility for the server that runs on the local system.

**cdsdel**
Deletes recursively the namespace entries of a cell.

**cdsli** Lists recursively the namespace entries of a cell.

– RPC Commands:

**rpcprotseqs**
Determines the supported protocol on a given host.

**rpcresolve**
Recursively resolves the elements of a namespace entry.

– Security Commands:

**rmxcred**
Purge expired tickets from the credentials directory.

## Standards Conformance

- **Standards Conformance Highlights** DCE 3.1 for Solaris supports the standards listed below, but cannot claim conformance to these standards because some of them are not in final form or because conformance tests do not exist.

**Threads**

– POSIX 1003.4a, draft 4

– AES/Distributed Computing - Threads

**RPC** AES/Distributed Computing - Remote Procedure Call

**Security**

– Authentication

- Kerberos Version 5, draft 4

– Authorization

- POSIX 1003.6, draft 12 (acls)

– AES/Distributed Computing - Security

– GSSAPI, including Internet RFC 1964

**Directory**

– AES/Distributed Computing - Directory Services

– X/OPEN-X.400 API Association XDS API Draft 6

**Transport Glue**

– RFC 1006, TPO-to-TCP

**Time**

– RFC 1129, NTP

– AES/Distributed Computing - Directory Services

## Compatibility with Solaris

This section describes the compatibility of DCE for Solaris with the supported versions of Solaris.

- The **man** command is *not* supported to display current DCE reference documentation.
- Solaris Network Computing System (NCS) version 1.5.1 and the DCE Base Services for Solaris can coexist on the same system because the DCE **dced** process provides the functionality that NCS applications expect from the **llbd** command.
- **Debugging Multi-Threaded Applications**: The Solaris **dbx** debugging command has the capability to recognize and debug multiple threads. For more information on the debugger, see the *IBM DCE Version 3.1 for Solaris: Application Development Guide—Core Components* .
- **C++ and DCE Compatibility**: The following discusses C++ and DCE compatibility.
  - **Compiling and Linking**: Using **C++** with DCE requires a few considerations, but generally nothing beyond what is required in using a C based library with C++.
  - **DCE Exceptions**: DCE exceptions are separate from the exceptions that the C++ language specification provides. The primary limitation, in using DCE exceptions within C++ programs is that, when a DCE exception is raised, destructors will not be called as the stack is unwound. The programmer must make sure that the objects are freed explicitly when DCE exceptions are handled. This might eliminate the use of automatically allocated objects within segments of the application code.
  - **C/C++ Interaction**: Again, as with any C functions called from C++, be sure to include DCE header files in external C declarations. This makes sure that the C++ linkage looks for the non-mangled C names, not C++ names.

    In C, memory is typically allocated using malloc. In C++, memory is allocated using new *object_type*. DCE adds **rpc_ss_allocate** for volatile data that needs to be freed by the system after an rpc returns. Care needs to be taken to make sure that memory allocated by one method is always freed using the corresponding routine.

As with any C library used in C++, it can be difficult to maintain a *pure* object-oriented architecture. In many cases, the components in DCE are fairly object-oriented in design, but since most of the pieces of DCE are designed to work together, they often pass data structures between mostly unrelated functions. For example, a login handle is an opaque data type that has a core of several closely related functions to manage and maintain the login context. While this lends itself well to grouping the data and functions as an object, the handle will need to be passed either implicitly or explicitly to most other objects that might be created. Since it is bad form to expose a data value inside an object, a sophisticated design needs to be considered (possibly a handle or surrogate object).

## Unsupported OSF DCE Features

The differences are grouped into sections by type. Each section is further subdivided into functional categories, which correspond with specific DCE services (such as configuration, security, and Cell Directory Services).

**Unsupported Services**:

- Security:
  - Transitive Trust in a cell hierarchy.
  - The Public Key Certificate Management API.
  - The Private Key Storage server.
  - Public key login using the OSF DCE 1.2.2 protocol has been superseded by the public key certificate login protocol. However, the security server can still process login requests from other DCE clients that support the OSF DCE 1.2.2 public key login protocol.
- Directory:
  - Hierarchical Cells.
  - Global Directory Services (GDS) are not provided in this release. However, GDS can exist in the same cell and be used for intercell communications, if it is provided by another product and if the cell adheres to the X.500 naming convention.
- dcecp (DCE Control Program):
  - **host configure**–Configures a host into the cell as a client or server.
  - **host unconfigure**–Removes the host from the name and security databases.
  - **host start**–Starts DCE on the specified host.
  - **host stop**–Stops DCE on the specified host.
- RPC:

– Single threaded RPC.

**Unsupported Commands**:

- **cdsbrowser**
- Configuration:

  The **dce_config** script has been replaced by other configuration commands. See the *IBM DCE Version 3.1 for Solaris: Administration Commands Reference* for more information.
- Security:

  The **sec_salvage_db**, **rlogin**, **rlogind**, **rsh**, and **rshd** commands supplied by OSF.
- Distributed Time Service:

  The **dtss-graph** command, which converts synch trace to PostScript.

## Limitations of Supported Services

There are several limitations for accounts configured to use Public Key authentication. These include:

- Public Key accounts cannot use the Password Strength Server.
- The key management API is for use only by applications using the shared-secret key authentication protocol. Applications using public key accounts must use the user-to-user protocol.
- When using GSSAPI, the DCE administrator must set up an account in the DCE registry database for the initiator and the acceptor. The following restrictions apply to the account for the acceptor:
  – The account for the acceptor must be set up to use a key in a keytab file as the account's password.
  – The account for the acceptor cannot be set up to use the user-to-user protocol.
  – The account for the acceptor cannot be set up to use the public key authentication protocol.

  No restrictions apply to the account for the initiator.

# Part 2. Planning for and Installing DCE 3.1 for Solaris

# Chapter 2. Planning

## System Requirements

All contents in the DCE 3.1 for Solaris product require the retail release of Sun Solaris 7. See the **README** for updates to the listing of supported versions of Solaris.

## Disk Space Requirements

See *IBM DCE Version 3.1 for Solaris: Release Notes* for the most current package disk space requirements.

## Global and Cell Considerations

The purpose of this section is to assist you in planning for the installation and configuration of DCE. DCE provides configuration utilities to assist you. "Chapter 3. Installing DCE 3.1 for Solaris Servers and Clients" on page 43 and "Configuring DCE" on page 55 describe the configuration process, including installing executable files, setting up a DCE cell, and configuring servers and clients.

This section discusses the following topics:



### Planning Questions to Consider

You need to consider a number of questions when planning for a distributed system.

Keep in mind the following global considerations as you plan for DCE:

- How much do you think your environment will grow in the next few years? Do you anticipate rapid or relatively slow expansion of your network?

If you think your environment will grow rapidly, consider setting up several cells representing smaller units of your organization. You can manage these smaller units as your network expands. As explained previously, members of each cell share a common purpose, and the cell is a unit of administration and security. If you anticipate slow expansion of your network, you might be able to establish one or more cells based on the organization that exists now. Consider how many administrators you will need to maintain your DCE cell, based on anticipated future growth.

- How much information updating do you require? Do the users in your network mainly look up information, or do they create and change information at their workstations?

  If information changes frequently and users in your network depend on the accuracy of that information, you need to consider how much you rely on replication. It is better to go to a central source of information for data that changes frequently. If users look up information, but do not need to change the information that is shared with other users, you can rely more on replicated data.

- Is the most important data the most available? Have you made plans to replicate this data?

  CDS and the Security Service maintain master copies of their respective databases. Each CDS directory can be replicated separately. The security service supports replication of the entire registry database. Because other components depend on the information managed by the Security Service and parts of the CDS namespace, that data needs to be available at all times. For example, the special character string **/.:** (the cell root) is stored in CDS and must always be available.

  Keep in mind that while replicating data helps availability, there is a cost in terms of performance and the amount of administration required.

- If your network has a gateway, are servers located on the same side of the gateway as the clients that rely on those servers?

  CDS servers broadcast messages at regular intervals to advertise their existence to CDS clients in the network. Clients learn about servers by listening for these advertisements. Placing servers and the clients that rely on them on the same side of the gateway facilitates efficient updates of information and a quick response to client requests. Additional administration is required if you rely on servers that are not available through the advertisement protocol, which is effective only in a local area network.

  On a LAN that has no CDS servers, proxy advertisers will broadcast the addresses of CDS servers. This means that clients do not need to know the address of a CDS server at the time of configuration. The proxy advertiser will broadcast the address of the CDS server that it was configured with. Additional CDS server addresses can be added using either the **cdscp define server** command or the **dcecp cdscache create** command.

Consider how fast and how expensive links are if you are administering a cell that includes users in different geographic locations. You might want to keep more information locally to reduce your dependence on transmitting information across links.

- Is communication limited to your own cell, or do you need to communicate with other cells?

  For your cell to communicate with other cells, you must:
  - Establish a unique Domain Naming Service global name for your cell
  - Define your cell in DNS
  - Have at least one GDA in your cell or have performed a **cdscp define server** or a **dcecp cdscache create**.

  **Note:** Global Directory Service (GDS) is not provided with this release of DCE 3.1 for Solaris. However, this release can use GDS if it is provided by another product to locate other cells.

  You can set up a special account in your cell's security registry for a foreign cell, indicating that your cell trusts the Authentication Service of the other cell, and a special account in the foreign cell's security registry to represent your cell. (For information about setting up these special accounts, see the *IBM DCE Version 3.1 for Solaris: Administration Guide*.) Even if you do not need to communicate with other cells now, consider whether you will need to communicate with other cells in the future. Be sure to establish a cell name with these future requirements in mind.

Your answers to these questions determine the basic requirements of your user environment. Use these requirements to help you decide on the optimum use of the DCE functions described in this and the following sections.

- **Resolving Differences between DCE and Solaris Standard Accounts**:

  It is strongly recommended that any users and groups defined in the individual system **/etc/passwd** and **/etc/group** files be synchronized with users and groups in the DCE registry. Synchronization can be facilitated with the **passwd_export** and **passwd_import** utilities after initial cell configuration. Any users who are not synchronized between the cell registry and the local files might not realize full benefit of the integration feature. On the other hand, this flexible integration scheme supports wandering users (users who are defined in the DCE registry, but not a local system). If a machine is configured to allow it, those wandering users may log onto the system and obtain DCE credentials and local access based on UNIX-relevant information in the registry.

  When DCE creates the security registry database, DCE includes some standard UNIX principals, groups, and accounts. These do not match those that are included on a typical Solaris system. This mismatch can lead to problems if you plan to use the **passwd_export** command to keep **/etc/passwd** and **/etc/group** synchronized with the DCE registry.

If you will include only Solaris machines in your cell, you can delete the standard principals, groups, and accounts from the registry and add those that match Solaris principals, groups, and accounts.

If your cell will include more types of machines than Solaris machines, you can either convert the standard accounts as described in the preceding paragraph or keep the accounts that DCE creates. Then, you can use the **/opt/dcelocal/etc/passwd_override** and **/opt/dcelocal/etc/group_override** files on individual machines to set up standard accounts and groups that match those expected by that machine's operating system. For more information about the override files, see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components*

If you plan to convert standard UNIX accounts in the registry as described here, you should do so immediately after initial cell configuration to reduce the likelihood of producing orphans (objects owned by UUIDs that have been deleted).

- **DCE Application Core Files**: Because DCE applications are multithreaded, their core files become large. Each thread has its own stack and other associated information that are saved in the core file. If you want usable core files from your DCE application while you are developing and testing the applications, make sure you have permission to write large core files. You can use the **ulimit** command to temporarily change the maximum core file size for the current shell process.

## Establishing a Cell Name

Before you can configure your DCE cell, you need to establish a cell name. This section describes DCE naming syntax, naming conventions, and the procedure for obtaining a cell name.

### Global Names

All DCE objects, including applications, machines, and users, have a global name. A global name is meaningful and usable from anywhere in the DCE environment. In DCE, global names begin with the special character string **/...**, which indicates the global root directory.

**DNS Global Names:** DCE also supports global directory operations through the use of DNS. Following is an example of a global name that uses the DNS format:

**/.../seattle.xyz.com/sec/principal/smith**

In DNS format, **/.../seattle.xyz.com** is the cell name, followed by a cell namespace entry.

### Cell-Relative Names

In the two previous examples, **sec/principal/smith** is that part of the global name that resides in the local cell. The **sec/principal/smith** part of the global name can be used to construct a cell-relative name. Cell-relative names, also known as local names, are meaningful only from within the cell where the name entry exists. Cell-relative names begin with the special character string /.:, which replaces the global part of the name (the cell name). If you are in the **seattle.xyz.com** cell, the following cell-relative name translates to the same global name shown in the previous examples:

**/.:/sec/principal/smith**

When you are entering a CDS name from the cell where that object is registered, you can use the cell-relative name. However, if you are entering a CDS name from another cell, you must use the global name, beginning with the character string **/...** (the global root).

CDS and DNS naming conventions are described in more detail in the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components*.

### Choosing a DCE Cell Name

Choosing an appropriate DCE cell name is important for the following reasons:

- DCE cells that will ever participate in the global namespace must have unique names to differentiate them from cells in other organizations.
- A uniquely identified cell name is critical to the operation of DCE security; this name is the basis for authentication in your cell.
- DNS expects global cell names to have a certain format. Choose a name that conforms to DNS naming conventions.
- DCE does not currently support cells registered simultaneously in GDS and DNS.

Note that cell names are case insensitive; that is, the name **MyCell** is equivalent to the name **MYCELL**. (When comparing cell names, DCE routines change the names to all lowercase before making the comparison.)

Cell names must not contain an at sign (@). Two cells on the same LAN should not have the same name. Two cells with the same name would be perceived as a single cell. Depending upon configuration, this could cause serious disruptions in the proper functioning of the commonly named cells. Cell names must also be restricted to characters in the DCE Portable Character Set.

### Obtaining a DCE Cell Name

If you plan to create a private cell and do not ever intend for it to communicate with cells outside your organization, you are not required to obtain a globally unique cell name. However, in order for your cell to communicate with other cells outside your organization, you need to have intercell set up and, before you configure your cell, you need to obtain a globally unique cell name from the GDS or DNS global naming authorities. The name can be one that already exists and is in use, or you can specify that you need a new name. This registration must be completed before you begin to configure the cell namespace. It is recommended that you obtain a unique global name for your cell even if you do not initially use a global directory service to communicate with other cells so that you can do so in the future.

### Defining a Cell in DNS

You can use the **cdscp** subcommand **show cell** to obtain data that you need to create or modify a cell entry in DNS. The data you obtain from the command is what CDS uses to contact servers in foreign cells. Use the **mkreg.dce** command to register cell information with the DNS. For information on setting up the intercell environment, managing intercell naming, and administering a multicell environment, see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components* .

## The Cell Namespace

An integral part of planning for a DCE cell is understanding the organization of your cell namespace. Consider the following as you plan the organization of a cell in your network:

- Are security requirements maintained?
- Does the organization of the cell facilitate network traffic where data sharing needs are the greatest?
- How will you manage the administrative accounts created for each DCE service during the configuration process?

### Determining Cell Boundaries

In DCE, the boundaries of a cell are equivalent to the boundaries of the cell namespace. A small organization can consist of one cell. A large organization can have many cells. The primary factors in determining a cell's boundaries are the common purpose and trust shared by the cell's principals. Principals within a cell can belong to groups that share the same privileges. Members of a group share the same level of trust and are authorized to perform certain actions.

Because there is a set of administrative tasks associated with setting up and maintaining each cell, it is reasonable to keep the number of cells in your organization to a minimum. However, the level of trust shared by groups of principals is a more important consideration than administrative overhead.

### Keeping Cells Stable

Once you decide how many cells you need and where the boundaries of those cells will be, make an effort to keep your cell structure stable. Servers are not easily moved from one cell to another; so, be sure to plan your namespace structure carefully in order to minimize reconfiguration. If you do need to move a host from one cell to another, you must:

- Move server processes from the host.
- Unconfigure the host from the old cell, using the **unconfig.dce** command.
- Use the **config.dce** command to reconfigure the host in the new cell.

### Types of Cell Namespace Entries

This section describes the different types of entries that comprise the cell namespace. These entries are created when you follow the default configuration path described in Configuring DCE. The cell namespace can be divided into the following parts:

- The CDS part of the namespace
- The security part of the namespace
- The DFS part of the namespace (the filespace)
- The **dced** (per host) part of the namespace

Each DCE service maintains its own namespace within the DCE cell namespace. DFS maintains its own namespace to ensure consistency among many files.The security service maintains its own namespace to ensure that the DCE cell remains secure. Clients of this service query CDS for binding information that enables them to find the security server. Certain entries in the CDS namespace contain binding information that enables a client to connect to a server outside of the Directory Service. One such entry connects clients to the part of the cell namespace that the security service manages. This entry or transition point between two namespaces is called a junction. The **/.:/sec** directory is the junction from the CDS part to the security part of the cell namespace, and the **/.:/fs** directory is the junction from the CDS part to the DFS part of the cell namespace.

The junction **/.:/hosts/***hostname***/config** is the junction from CDS to the **dced** (per host) part of the namespace.

Figure 1 on page 24 shows the top level of the cell namespace. In some cases, the names in the cell namespace are fixed (or well known) and cannot be

changed. In other cases, you can choose a different name from the one listed. In Figure 1, **/.:** and **cell-profile** are well-known names.

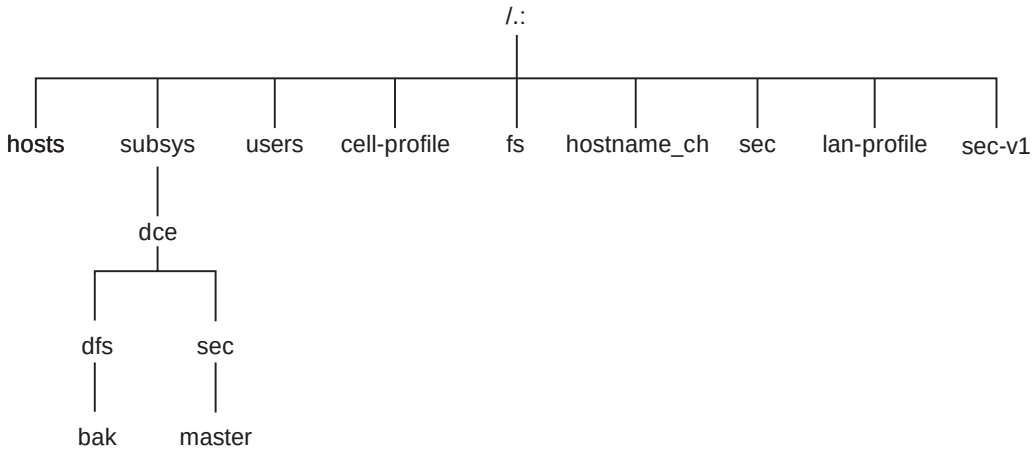


*Figure 1. Top Level of the Cell Namespace*

You can use the **dcecp**, **rpccp**, **cdscp**, or **cdsli** commands to view the CDS namespace, including the **sec** and **fs** junctions. You can use commands such as **dcecp** to see the contents of the security portion of the CDS namespace.

**CDS Namespace Entries:** The DCE Cell Directory Service is a distributed, replicated database service that is used to store names and attributes of resources located in a DCE cell. This database consists of a hierarchical set of names called the namespace. Each CDS server maintains a portion of the namespace in a local database called a *clearinghouse*, which is optimized for local access. A clearinghouse is designed for relatively few **write** operations (such as creating or deleting directories and objects or exporting binding information), but many **read** operations (such as importing binding information). Note that a clearinghouse is automatically created during the configuration process for a CDS server. See "Configuring the Initial CDS Server" on page 62 for more information.

A CDS database that is distributed and replicated among multiple CDS servers and multiple clearinghouses must be kept consistent. The large number of **write** operations used to replicate and maintain consistency can cause stress-induced CDS failures. Using a large number of replicated CDS directories can also result in stress because updates must be propagated to all the read-only replicas. Use the **cdscp show server** command to display the number of **read** and **write** operations handled by a server since the service was started. This command allows you to monitor the level of activity and adjust the configuration if necessary.

As a directory service, CDS is designed to manage information that does not change often. For example, binding information stored in CDS does not include endpoints since endpoints change frequently. As you design applications, avoid the need to store highly dynamic data in the CDS namespace.

The CDS namespace contains entries for servers, hosts, CDS clearinghouses (collections of directory replicas stored at a particular server), RPC profiles, RPC groups, and subsystems. The entries have a CDS type of *directory* or *object*, indicating the kind of CDS entry to which the name refers. A CDS directory is a container in which objects are stored. CDS uses directories to organize groups of object entries.

Profiles catalogued in the CDS namespace specify a search path through the Directory Service. The cell profile (**/.:/cell-profile**) stores the location of the servers that are available in the cell, regardless of physical location. In a geographically dispersed cell, servers can be located in different cities or even different countries. The LAN profile defines alternate servers that can be used in situations where geographic proximity is important. For example, **/.:/lan-profile** is the default LAN profile used by DTS. This profile contains entries for the DTS server local set. If a cell spans more than one LAN, a profile can be created for each LAN that the cell spans. For example, in a cell that encompasses two LANs, you can direct hosts on one LAN to **/.:/lanA-profile** and hosts on the other LAN to **/.:/lanB-profile**. For information on setting up multiple LAN profiles, see "Configuring DCE" on page 55.

**Security Namespace Entries:** The types of security entries are as follows:

**principal**
: This type of entry contains an individual principal.

**principal directory**
: This type of entry contains individual principals or one or more principal directories, or both.

**group** This type of entry contains an individual group.

**group directory**
: This type of entry contains individual groups or one or more group directories, or both.

**org** This type of entry contains an individual organization.

**org directory**
: This type of entry contains individual organizations or one or more organization directories, or both.

**policy** This type of entry contains a security policy.

When you (or an application) are accessing an entry in the security part of the namespace, the name of the entry alone provides enough information for the security service to work with. For example, the security server knows that the login name is a principal name, registered in the security part of the namespace; **/.:/***principal_name*, **/.../***cell_name/principal_name*, and *principal_name* are all valid ways of representing the name you use to log in.

When you use the **dcecp** command, you specify the type of object you will operate on. For example, to change account information associated with the principal **smith**, you specify that you want to operate on an account. You then enter the principal name **smith**. The **dcecp** command deals with the following types of objects related to security:

- Principals
- Groups
- Organizations
- Accounts
- Xattrschemas

The *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components* explains how to use the **dcecp** command to display information related to principals, groups, organizations, accounts, and xattrschemas.

The **dcecp** also supports operations performed by **acl_edit**. The acl object of **dcecp** is used for this purpose. The **dcecp** command requires the object's fully qualified path name when modifying acls, as shown in the following example:

```
/.:/sec/principal/smith
```

and not simply the following:

```
smith
```

The following parts of the namespace comprise the security namespace:

**/.:/sec/principal**
**/.:/sec/group**
**/.:/sec/org**
**/.:/sec/policy**
**/.:/sec/xattrschema**

### CDS Namespace Replication Considerations

Directory replication is the most reliable way to back up the information in your CDS namespace. Because the CDS data is replicated by directory, when you replicate a directory, all of the object's entries in it are automatically replicated. Use the **dcecp** control program to create replicas of directories at a

CDS clearinghouse. If you create a clearinghouse in addition to those that are automatically created on a CDS Server at configuration time, that clearinghouse must be created in the root directory (**/.:**) of the cell namespace.

Follow these guidelines for replicating parts of the cell namespace:

- The root directory (**/.:**) is automatically replicated (without child directories) when you create a clearinghouse.
- You should have at least two replicas of each CDS directory to ensure the entire namespace is available at all times. For further information about backing up CDS information, see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components*.

## Planning for Access Control

In planning for access control, it is important to keep the level of access control in your cell restrictive enough to ensure that security is maintained. A special set of individuals or a special group can be given permission to create accounts and groups in the root directory of the security namespace. The **acct-admin** group is created when you configure DCE. **acct-admin** is the only group that can create accounts and groups in the root directory of the security space.

While maintaining an adequate level of security in your cell, you also need to consider the requirements of administrators who are maintaining DCE services when you set access control levels.

Following are some of the groups created when you configure DCE using the **config.dce** command:

**sec-admin**
: This group administers security servers, cell registry functions, and other security functions.

**audit-admin**
: This group administers Audit servers and related audit functions.

**cds-admin**
: This group administers CDS servers, CDS replication, and other CDS functions.

**dced-admin**
: This group administers DCE host servers and ACLs.

**dts-admin**
: This group administers DTS servers and related DTS functions.

**dfs-admin**
: This group administers DFS File Servers and related DFS functions.

In addition to the administrative groups, individual users need permission to control some information kept in the registry database. For example, a user needs to be able to change its password, home directory, or login shell.

## DCE Naming Considerations for Internationalization

Standard (OSF) DCE, restricts entries in the security namespace, such as principal names, to the characters in the DCE Portable Character Set. See the Architectural Overview of DCE in the *IBM DCE Version 3.1 for Solaris: Introduction to DCE* for the definition of the DCE Portable Character Set. IBM DCE provides an override capability which enables the use of non-portable characters.

This capability should be used only in environments that are homogeneous with respect to code set and only in environments in which all DCE installations support this extension. Security namespace entries that use non-portable characters are guaranteed to work correctly only when the code set of the entire enterprise is the same as that of the process under which the names were created. To enable the use of non-portable security names, the environment variable DCE_USE_NONPORTABLE_NAMES must be set to 1 before DCE is started, in all client and server processes in which DCE Security will run.

Certain other names, such as CDS directory names, can also be composed of characters from outside of the DCE Portable Character Set. Because DCE does not perform code set conversion on names, non-portable characters should be used only in environments which are, and will remain, homogeneous with respect to the code set. In environments which are not homogeneous with respect to code set, all DCE names must be restricted to the DCE Portable Code Set.

Subject to the previously mentioned restrictions and to the additional naming rules documented in the *IBM DCE Version 3.1 for Solaris: Introduction to DCE* and the *IBM DCE Version 3.1 for Solaris: Application Development Guide—Core Components*, the following names can contain characters outside of the Portable Character Set:

- CDS Object
- CDS Directory
- CDS Attribute
- CDS Link
- RPC idl_byte data
- RPC full name
- Principal
- Group

- Organization
- ERA

## Client and Server Considerations

This section describes configurations for DCE client machines, the different types of DCE server machines and DCE Application Development Environment machines. A DCE client machine can run client code of every DCE service. DCE server machines are configured to run a certain set of software. This software is made up of at least one daemon and, in some cases, one or more additional programs that comprise the server side of a DCE component. DCE server machines also run the software that makes up the DCE client configuration.

The following topics are provided:
- "Determining Requirements for DCE Client Machines"
- "Determining Requirements for DCE Server Machines" on page 33
- "DCE Administration Utilities" on page 36.

### Determining Requirements for DCE Client Machines

This section describes the planning considerations involved in setting up DCE client machines. All DCE machines, including DCE server machines, are also DCE clients.

The following subsections describe the executables that run on a DCE client machine.

#### RPC Client Programs

A DCE client contains the following RPC programs:

- The **dced** daemon must run on any machine that has an RPC server process that exports an interface with dynamic bindings. The **dced** daemon is used to register binding information.

  The **dced** daemon must be running before you configure any other DCE services that register their endpoints. DCE services need to register their endpoints with **dced**. Only one **dced** daemon can run on a machine at a time, because **dced** uses a well-known port.

  Network interfaces, routing services, and other network services must be available before RPC starts. The **dced** daemon is started by the **start.dce** command. The **start.dce** command can be invoked during boot by specifying the **-autostart yes** option on the **config.dce** command or by

adding the symlink: **/etc/rc3.d/S15–20dce** to **/etc/init.d/dce**. This will allow DCE services to be brought up each time the machine boots.

- The DCE control program (**dcecp**) is a utility that allows you to browse, update, add, and delete the RPC attributes of entries stored in the CDS namespace and the endpoints that are managed by local and remote **dced** daemons.

### Security Service Client Programs

The **dced** daemon maintains the local machine principal identity by periodically refreshing the ticket-granting ticket for the machine's principal. This assures that the local root user or any daemon who inherits the machine identity has valid DCE credentials. The **dced** daemon also exports and implements a variety of interfaces, including password and group override support, certification of the security server, and pre-authentication support.

For more information about ticket-granting tickets, see *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components* .

### CDS Client Programs

The DCE client runs the following CDS processes:

- The CDS advertiser, the **cdsadv** process, allows applications to access and communicate with **cdsd**. It starts any needed CDS clerks (**cdsclerk**) and creates the cache shared by the local CDS clerks.
- The **cdsclerk** is an interface between CDS client applications and CDS servers. A clerk must exist on every machine that runs a CDS client application. One **cdsclerk** process runs for each Solaris principal on a machine that accesses CDS. The CDS clerk handles requests from client applications to a server and caches the results returned by the server. Because results of the server request are cached, the clerk does not have to go repeatedly to the server for the same information. All CDS clerks on a machine share one cache. One clerk can serve many client applications.
- The DCE control program (**dcecp**) can be used to browse, update and delete CDS entries, and manage the namespace. For more information, see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components*.
- The CDS control program, **cdscp**, is a command interface used to control CDS servers and clerks and manage the namespace and its contents. The **cdscp** command interface was available with previous versions of DCE and is provided to ease migration to the use of the **dcecp** utility. For more information about the CDS control program, see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components*.

### DTS Client Programs

The DCE client runs the following DTS processes:

- The **dtsd** daemon is set as a client or a server. On a client machine, **dtsd** synchronizes the local clock.
- The **dtscp** program allows you to administer DTS, including configuring the **dtsd** daemon as either a client or a server.

### Slim Client Programs

In general, client systems have less available memory than server systems. If a client does not offer DCE services to other systems in the cell, it might not need all of the functions provided by the daemons started by the configuration of DCE software on the client system. The slim client offers the capability of being such a ″reduced″ client in the cell.

Since no information about the slim client is kept in the cell, administrator intervention, that is **cell_admin**, is not required to configure it. Use the **config.dce** command to configure the slim client. Use the **start.dce** and **stop.dce** commands respectively to start and stop the slim client. To unconfigure the slim client use the **unconfig.dce** command.

The slim client option reduces DCE memory consumption on client systems by running a single instance of the CDS clerk with no other DCE daemons. Running a single instance of the CDS clerk is done by starting the clerk with the **-n** option. This starts a clerk without the CDS advertiser. However, if there are so many other DCE services and functions that can be run, how can a single CDS clerk be sufficient? The answer is that most DCE clients need only the following DCE functions:

- RPC calls (both authenticated and unauthenticated)
- DCE login
- CDS name lookups

For RPC calls and most logins, no DCE daemons are needed. These functions simply use RPC runtime routines and security runtime routines.

For CDS name lookups, only a CDS clerk is necessary. With full DCE, CDS clerks are started by the CDS advertiser, requiring a CDS advertiser to be present. However, in DCE 3.1 for Solaris, the **-n** option on the **cdsclerk** command starts a single instance of the CDS clerk without needing the advertiser. This clerk will not terminate after being idle for 20 minutes, as it does in full DCE. Additionally, when the clerk is started in this fashion, it takes over the role of the CDS advertiser in managing the CDS client cache.

Without an advertiser, the **cdsclerk** can not be managed by **dcecp** or **cdscp**. The following commands will fail:

```
cdscp show clerk
cdscp disable clerk
cdscp show cached clearinghouse
cdscp define cached server
cdscp show cached server
cdscp clear cached server
dcecp -c cdscache create
dcecp -c cdscache delete
dcecp -c cdscache show -server
dcecp -c cdscache show -clearinghouse
```

The following **dced** services do not run on a DCE slim client:

- **dced Endpoint Mapper Service** must run on any system providing a service that can be accessed through Remote Procedure Calls (RPCs). Such a server is called an RPC server. When a system issues an RPC to an RPC service, it uses the RPC runtime routines to send the request to a specific machine address and asks for the desired RPC service by name. After the RPC reaches the machine where the service resides, the Endpoint Mapper Service maps the RPC service name to the endpoint, or port number, of the specific program providing the service. After the endpoint is known, the client is bound to the specific RPC service and RPCs can be issued directly to that service.

  Although every DCE client system issues RPCs, most do not need the Endpoint Mapper Service, because they are probably not RPC servers. Therefore, the RPC-related limitation of not running **dced** on a client system is that it cannot be an RPC server.
- **Security Validation Service** provides the functions listed below. If a client system does not need these functions, it does not need the **dced** Security Validation Service.
  - **Security Server Certification**
  - **Third-party pre-authentication during dce_login**
  - **Keeping the machine context up to date**.
  - **Password and group overrides**.
- **Preferred Security Replica** is not supported for the slim client.
- **System Management Services**:

  The system management functions provided by DCE are listed below. Without **dced**, a client system cannot be remotely managed by means of these functions.
  - **Host Data Management**: This service maintains local files of host data (that includes the host name, cell name, and cell aliases) and a post-processor file. The post-processor file contains program names that are matched to other host data items. **dced** runs the program if the corresponding host data item changes.

- **Server Control**: This service maintains data that describes the startup configuration and execution state for each server. It can also start or stop particular servers, and enable or disable specific services of servers. This service is not needed on a client that is not running any RPC servers.
- **Key Table Management**: This service allows for the remote maintenance of a server's key tables. This service is not needed on a client that is not running any RPC servers.

**Security Integration (PAM and NSS)** can run on a slim client. Be aware that because the certification service is not available, when a user logs in, the user's identity cannot be certified to have been issued by a legitimate security server and that security integration on a slim client cannot use password and group overrides. Because the machine context is not available, security integration on a slim client uses unauthenticated access to the registry.

## Determining Requirements for DCE Server Machines

This section provides information about requirements for the different types of DCE server machines.

### Files Installed on DCE Server Machines

The following subsections discuss the files that must be installed on each of the different DCE server machines and the approximate space required. Note that, because all DCE servers are also DCE clients, the files described in "Determining Requirements for DCE Client Machines" on page 29 must also be installed on server machines. Therefore, add the appropriate server space requirements to the DCE client machine space requirements to reach the approximate total space requirement for the configuration you are planning.

### Security Server Processes

Every cell has one master DCE security service machine and can also have replica DCE security service machines. The following processes run on a DCE security service master or replica server machine:

- The security server, or **secd** process, implements the Authentication service, the Privilege service, and the Registry service.
- The **sec_create_db** program initializes the security database. The **config.dce** command passes a parameter indicating whether to create a master or replica security server on the machine.
- The DCE control program (**dcecp**) is used for the registry, management, and maintenance of the security server. Optionally, you can use the **sec_admin** program. See "DCE Administration Utilities" on page 36 for descriptions of these programs.

Keep the following considerations in mind when you are planning for security servers:

- The node that runs the master security server must be highly available and physically secure. Consider placing the master security server machine in a locked room and keeping a log to record who accesses the machine.
- Be sure to move the master security server before removing the node from the network or shutting down the node for an extended period of time. Modifications are made to the master security server and propagated to replicas throughout your cell. If the master security server is unavailable, no updates can be made. For more information see ″Handling Network Reconfigurations″ in the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components*.
- A cell can have only one master security server. If you plan to make one cell out of several existing cells with independent master security servers, you must first merge their registries.
- Keep the following considerations in mind when you are planning for identity mapping servers:
  - The DCE security server uses the identity mapping server when a DCE user logs in using the public key certificate login feature. You must configure at least one identity mapping server to use DCE public key certificate login.
  - Because of the interaction between the identity mapping server and the DCE security server, nodes which run identity mapping servers should be highly available and physically secure. It is recommended that an identity mapping server be run on each node which runs a DCE security server.

For further information about planning for the DCE security service, see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components*.

Since the security registry is replicated in full across all security replicas, if the host that contains the master security server goes down, hosts that have replica DCE security servers can still provide registry information; so, consider having a number of replicas in your network. Use factors such as the number of machines in your cell, the reliability of the machines that run security servers, and your cell's available resources to determine how many replica security servers you need to have.

### CDS and GDA Server Processes

A CDS server stores and maintains object names within a cell and handles requests to create, modify, and look up data. A GDA server enables the cell in which it is running to communicate with other cells.

The following processes run on a CDS server machine:

- The CDS daemon, **cdsd**, is the CDS server process.
- The **cdsadv**, in addition to receiving server advertisements to find out what servers are available as it does on a DCE client machine, on a CDS Server machine also sends server advertisements.
- The DCE control program (**dcecp**) for the management and maintenance of the CDS software. In addition, the **cdscp** program for controlling and displaying information about CDS clerks and servers. See "DCE Administration Utilities" on page 36 for descriptions of these programs.

In preparing for CDS, you need to select server nodes that store and maintain the clearinghouses (CDS databases) in the cell.

Keep the following guidelines in mind in order to achieve reliability, optimum performance, and data availability:

- Choose dependable nodes. A CDS server needs minimal downtime and needs to restart quickly. The CDS server needs to be one of the first systems available on the network because client applications and other DCE servers rely on the CDS server for up-to-date information. The CDS server initializes the CDS namespace when you configure DCE.
- Use reliable network connections. This helps to ensure that all servers maintaining directory replicas can be reached when CDS performs a skulk. Skulks are periodic updates that check for consistency across all replicas.
- Consider the size of your cell and how geographically dispersed the cell is when deciding how many CDS servers you need. You should have at least two copies (one master and one read-only replica) of each CDS directory to ensure access to data if one of the servers becomes unavailable.
- Each CDS server maintains at least one clearinghouse. All clearinghouses contain a copy of the root in addition to other directories replicated there.
- You need to make replication decisions based on where the contents of directories are referenced. Put replicas where the contents are read and put masters where the contents are written.

The **gdad** daemon is the GDA server, which sends lookup requests for cell names to the DNS and returns the results to the CDS clerk in the cell that initiated the request.

The GDA can be on the same machine as a CDS server, or it can exist independently on another machine. You can have two or more **gdad** daemons running in a cell to ensure GDA availability.

### DTS Server Programs

The DCE client configuration already contains all the files necessary for a DTS server machine, with the exception of the optional time provider.

- The **dtsd** daemon (which can be installed on a DCE client machine) is configured to run as a server. As a server process, **dtsd** synchronizes with other DTS servers, in addition to synchronizing the local clock, as it does on a client machine.
- The **dts_***device_name***_provider** specifies the communications between the DTS server process and the time-provider process. For *device_name*, substitute the device you are using, which can be a radio, clock, or modem, or another source of UTC time for DTS. A time provider is optional. If you use a time provider, it must connect to a server process.

Consider the following guidelines when planning your DTS implementation:

- Each cell should have at least three DTS servers. At least three DTS servers are needed in order to detect if one of them is faulty when they are queried for the time. It is preferable to have four or more DTS servers to provide redundancy. The additional servers increase the accuracy of time synchronization. However, increasing the number of servers queried for the time also increases the activity on the network. The administrator must balance the level of accuracy with the amount of network activity.
- A time provider is optional in DTS; however, cells that must be closely synchronized with a time standard need to have at least one time provider.
- Servers need to be located at the sites with the greatest number of different network connections.
- If there are less than three time servers configured in the cell, one of the following commands should be used:

  ```
  dtscp set servers required n
  (where n is the number of time servers in the cell)
  dcecp -c dts modify -minservers n
  (where n is the number of time servers in the cell)
  ```

  This will prevent a warning message from being logged every time the server attempts to sync.

There are many network configuration decisions that affect DTS planning. The *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components* contains details about the total DTS planning process, including configuration planning for Local Area Networks (LANs), extended LANs, and Wide Area Networks (WANs) as well as an explanation of the criteria you need to use when selecting a time source for your network to use.

## DCE Administration Utilities

This section describes the system administration utilities that can assist you in performing DCE administrative tasks.

### DCE Control Program

The DCE control program **dcecp** creates, maintains, and manages RPC, CDS, security, DTS, EMS, and DCED objects. For more information on **dcecp**, see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components* and the *IBM DCE Version 3.1 for Solaris: Administration Commands Reference* .

### RPC Administration Programs

The DCE Remote Procedure Call Service provides the following administration utilities:

- The **dced** daemon is used to register binding information.
- The DCE control program (**dcecp**) allows you to browse, update, add, and delete the RPC attributes of entries stored in the CDS namespace and the endpoints that are managed by local and remote **dced** daemons.

See the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components* and the *IBM DCE Version 3.1 for Solaris: Administration Commands Reference*for more detailed information about these programs.

### DCE Security Service Administration Programs

The DCE security service provides the following administration utilities:

- The **dcecp acl** command displays, adds, modifies, and deletes ACL entries for a specific object. The *IBM DCE Version 3.1 for Solaris: Administration Commands Reference* contains detailed information about using the **dcecp acl** command.
- The **dcecp account**, **group**, **organization**, **principal**, **registry**, **user**, and **xattraschema** commands allow you to edit the registry database or the local registry. Almost all editing of the registry database must be done with these commands. The *IBM DCE Version 3.1 for Solaris: Administration Commands Reference* explains the use of the commands.
- The **passwd_import** command allows you to create registry entries based on the group and password files from machines that do not implement DCE Security.
- The **passwd_export** command allows you to update the UNIX **/etc/passwd** and **/etc/group** files with current user information obtained from the registry.
- The **passwd_override** and **group_override** files allow you to establish overrides to the information contained in the registry.
- The **rmxcred** command purges expired tickets from the credentials directory.
- The **dcecp registry** command helps you manage server replicas of the registry, change the master server site, and reinitialize a subordinate server.

This command also helps you manage the security server and its database. You can perform tasks such as generating a new master key for the database and stopping the security server.

### CDS Administration Programs

CDS provides the following administration utilities:

- The **cdscp** program is described in "CDS Client Programs" on page 30.
- The **cdsli** gives a DCE user the ability to recursively list the namespace of cells.
- The **cdsdel** deletes recursively the namespace of cells.
- The DCE control program, **dcecp**, can be used to browse, update, and delete CDS entries, and to manage the namespace. It replaces **cdscp**.
- The **mkreg.dce** command enters information about your DCE cell into the database maintained by your domain name server (the **named** daemon).
- The **rmreg.dce** command removes information from the database that is maintained by your domain name server (the **named** daemon) that were added by the **mkreg.dce** command.

### SVC Administration Programs

The **svcdumplog** program prints the contents of a serviceability binary log file as readable txt. For more information on **svcdumplog**, see the *IBM DCE Version 3.1 for Solaris: Administration Commands Reference*. The **dce_err** program displays the text associated with a DCE message ID.

### DTS Administration Programs

- The **dtscp** command controls the interface you can use to configure and manage DTS. It is already included in the DCE client software.
- The **dtscp** program allows you to administer DTS, including configuring the **dtsd** daemon as either a client or a server.

### DCE Web Secure

The DCE Web Secure product extends your Netscape FastTrack 3.01 or Enterprise 3.61 Web server to provide DCE authentication to your Web transactions, enabling you to use a Web browser to run Common Gateway Interface (CGI) programs that require DCE credentials, such as DCE Administration.

With DCE Web Secure you can provide DCE credentials to a CGI program. For example, with a tcl program called changeusers.tcl that makes batch changes to a group of DCE accounts, administrators would place the changeusers.tcl program into a directory that has been configured for CGI access in the Web server configuration files. When a user runs the CGI

program through the Web browser, it gains the DCE credentials under the DCE userid that the Web browser user is logged in as.

## Application Development Environment

You can configure a DCE machine for the development of DCE applications. This configuration requires adding to the basic DCE client configuration several include (**.h**) and interface specification (**.idl**) files, along with the **idl** compiler. The files and the compiler are included in the **IDCEtools** package available in the DCE for Application Developers package. You can also use the **sams** utility to include support for messaging and serviceability in your applications. The **sams** utility is included in the **IDCEtools** package.

## Location of Installed DCE Files

The files used by DCE are grouped in the following locations:
- The **/opt/dcelocal** subdirectories
- Conventional UNIX subdirectories.

Some information needs to be kept locally on a machine for reliability and to ensure security is maintained. For example, when you configure DCE, the file that contains the name of your cell must be on the machine that is being configured. This file is stored in the **/opt/dcelocal** subtree.

The **/opt/dcelocal** subtree is created when you install DCE components.

Files are installed into the **/opt/dcelocal** subtree. Symbolic links are created from **/usr/lib**, **/usr/bin**, or **/bin** that can be used from the conventional UNIX subdirectories to **/opt/dcelocal**.

This section contains the following topics:


### The /opt/dcelocal Subtree

In order to initially boot a server and configure the cell, the appropriate files for mandatory servers (CDS and security) need to be available on that server machine (in the **/opt/dcelocal** subtree).

**Note:** It is strongly recommended that copies of the minimum set of programs and data files installed during the default DCE installation procedure be kept locally on server machines for stand-alone operation and emergency maintenance.

The contents of the **/opt/dcelocal** subtree can vary from machine to machine inside a DCE cell to accommodate and serve specific configurations. In addition, every machine must have local access to certain files so each machine can run as a stand-alone system if the machine is disconnected or partitioned from the cell. The appropriate files on DCE servers that have to be local to the server machine must be stored under **/opt/dcelocal**. Client-related data files are stored below **/opt/dcelocal/etc** (static configuration data) and **/opt/dcelocal/var/adm**. All server-specific data files are located in the **/opt/dcelocal/var/***dce-component-name* directory.

The **/opt/dcelocal** subtree is populated and initialized during DCE installation and configuration.

## Conventional UNIX Directories

Some DCE files and directories need to be accessible in conventional locations so users can conveniently access frequently used utilities and data. Symbolic links are created in conventional directories to the files and subdirectories in **/opt/dcelocal**. For example, idl is accessible via a symbolic link from the **/usr/bin** directory.

## File Locations

The installation process for DCE 3.1 for Solaris places files in the following locations:

**/opt/dce**
DCE binaries, machine-generic files.

**/opt/dcelocal**
Links to **/opt/dce** directories, local machine-specific files.

### File Systems to Create and Mount

You will probably want to create new Solaris file systems in order to use DCE effectively:

**/opt/dcelocal**
All DCE components store information in the **/opt/dcelocal** directory. If the **/opt** file system fills up, DCE and other subsystems that depend on **/opt** cannot operate correctly.

Before you install DCE, create a new file system mounted over **/opt/dcelocal/var**. Reserve at least 30 megabytes for **/opt/dcelocal/var** for your initial DCE configuration.

**/opt/dcelocal/var/directory**

The CDS server stores the clearinghouse files, which contain this server's portion of the namespace, and local data in this directory.

If this machine is configured as a CDS server, create a new file system mounted over **/opt/dcelocal/var/directory** before you install DCE.

Reserve about 30 megabytes for the server's use.

If you do not plan to create a separate files system for the CDS server, add the additional 30 megabytes to **/opt/dcelocal/var**.

**/opt/dcelocal/var/security**

This is where the security server stores the registry, credentials, and local data. If this machine will be a security server, you should add an additional 10 megabytes to **/opt/dcelocal/var** for the server's use.

Files stored in **/opt/dcelocal/var** are any files particular to the individual machine. You should monitor the space usage in **/opt/dcelocal/var** (and any associated separate files systems) to make sure it does not fill up. To clean up expired credentials files in **/opt/dcelocal/var**, use the **opt/dcelocal/bin/rmxcred** command. The DCE Auditing and Servicability facilities also use space in **/opt/dcelocal/var**. See the *IBM DCE Version 3.1 for Solaris: Administration Commands Reference* for more information on **rmxcred** and DCE Auditing. See the *IBM DCE Version 3.1 for Solaris: Problem Determination Guide* for more information on DCE serviceability logs.

# Chapter 3. Installing DCE 3.1 for Solaris Servers and Clients

Use the following sections for installation:
- "Installable Packages"
- "Installing DCE 3.1" on page 45

## Installable Packages

Following is a summary of the installable filesets for the DCE 3.1 for Solaris packages. For more detailed descriptions, see "Chapter 1. Overview of DCE 3.1 for Solaris" on page 3.

- IDCEpriv — DCE Privacy Level Protection feature
- IDCEcdss — DCE Cell Directory Services
- IDCEclnt — DCE Base Services
- IDCEenUSd — DCE U.S. English Online Documentation
- IDCEitd — DCE Italian Online Documentation
- IDCEkod — DCE Korean Online Documentation
- IDCEzhd — DCE Simplified Chinese (EUC) Online Documentation
- IDCEenUSm — DCE U.S. English Message Catalogs
- IDCEesm — DCE Spanish Message Catalogs
- IDCEitm — DCE Italian Message Catalogs
- IDCEjaJPm — DCE Japanese (PC Kanji) Message Catalogs
- IDCEjam — DCE Japanese (EUC) Message Catalogs
- IDCEkom — DCE Korean Message Catalogs
- IDCEZBKm — DCE Simiplified Chinese (GBK) Message Catalogs
- IDCEzhm — DCE Simplified Chinese (EUC) Message Catalogs
- IDCEsecs — DCE Security Services
- IDCEsmgmt — DCE System Management Services
- IDCEtools — DCE Tools

## Prerequisite Software

Table 1 on page 44 lists the DCE 3.1 for Solaris packages in the order in which they are installed. See the *IBM DCE Version 3.1 for Solaris: Release Notes* for the latest requisite levels of software.

**Note:** Those software names beginning with IDCE are at the same release level as the shipped DCE product.

*Table 1. Installation packages and prerequisite software*

| **Package** | **Prerequisite[1] Packages** | **Prerequisite Package Description** |
|---|---|---|
| IDCEclnt | Solaris 7 for SPARC | Solaris 7 for SPARC Operating System |
| IDCEsecs | IDCEclnt | DCE Client Services |
| IDCEcdss | IDCEclnt | DCE Client Services |
| IDCEtools | IDCEclnt | DCE Client Services |
| IDCEsmgmt | IDCEclnt | DCE Client Services |
| IDCEsmgmt | SUNWsasnm | Solstice Enterprise Agents Simple Network Management Protocol patch level 107709–01 |
| IDCEpriv | IDCEclnt | DCE Client Services |
| IDCEenUSm | IDCEclnt | DCE Client Services |
| IDCEitm[2] | IDCEclnt | DCE Client Services |
| IDCEesm[2] | IDCEclnt | DCE Client Services |
| IDCEjam[2] | IDCEclnt | DCE Client Services |
| IDCEjaJPm[2] | IDCEclnt | DCE Client Services |
| IDCEkom[2] | IDCEclnt | DCE Client Services |
| IDCEzhm[2] | IDCEclnt | DCE Client Services |
| IDCEZBKm[2] | IDCEclnt | DCE Client Services |
| IDCEenUSd | IDCEclnt | DCE Client Services |
| IDCEitd[3] | IDCEclnt | DCE Client Services |
| IDCEkod[3] | IDCEclnt | DCE Client Services |
| IDCEzhd[3] | IDCEclnt | DCE Client Services |

**Notes:**

[1]Prerequisite package(s) must be installed prior to the the package that you want to install. (The package can not be installed before the prerequisite package.)

[2]Translated Message Catalogs. Note that messages will be displayed in English unless the **NLSPATH** variable includes the clause **/usr/lib/locale/%L/LC_MESSAGES/%N**. See Chapter 3 of the *IBM DCE Version 3.1 for Solaris: Problem Determination Guide* for more information.

[3]Translated Online Documentation. Note that documentation will be displayed in English unless the **NLSPATH** variable includes the clause **/usr/lib/locale/%L/LC_MESSAGES/%N**. See Chapter 3 of the *IBM DCE Version 3.1 for Solaris: Problem Determination Guide* for more information.

## Installing DCE 3.1

If you are upgrading an existing installation, use the steps that are documented in "Migrating a Solaris DCE Cell to DCE 3.1 for Solaris".

If you are installing a system that does not currently have DCE installed, use **dcesetup** to install the appropriate DCE packages. The **dcesetup** command can be run with all information that is provided on the command line, or in prompting mode.

- Command line version:

```
dcesetup install -component {appdev | cdsserver | client | secserver | \
 sysmgmt | priv | msgs | docs}
-dir <DCE_release_directory> [-mklinks <DCE_target_directory>] [-force]
[-nomsgs] [-msg_langs {<en_US> | <it> | <es> | <ja> | ja_JP.PCK>| <ko> | \
 <zh> | <zh.GBK>}]
[-nodocs] [-doc_langs {<en_US> | <it> | <ko> | <zh>}]
```

- Prompting version:

```
dcesetup install
```

  You will be prompted for information.

## Migrating a Solaris DCE Cell to DCE 3.1 for Solaris

Because DCE 3.1 for Solaris is dependent upon Solaris 7 or higher, one of these versions must be installed on your machines to migrate from DCE 1.1 to DCE 3.1 for Solaris. You can do this migration without a reconfiguration of your existing DCE cell by using the following procedures. You are not required to migrate your machines in a specific order, but please pay close attention to the limitations on DCE security server functionality as described in **Migrating DCE Security Replicas** on page 47. Read this entire section before beginning the migration procedure.

- **Before Migrating:**
  1. In DCE 3.1 for Solaris, each workstation in a DCE cell keeps configuration information about the DCE clients and servers that run on the local machine. This information is stored locally in the **dced** server configuration database.

     During migration, the migration commands attempt to add entries for the currently configured servers to the configuration database. In order for the migration to succeed, however, the machine context (hosts/*dce_hostname*/self) requires the necessary permissions on the server configuration database to insert entries.

*Before you attempt to migrate a machine to DCE 3.1 for Solaris*, you must ensure that the machine context has control, read, insert, and insert-privileged permissions on the local machine's server configuration ACL

You can verify this by running the following command:

```
dcecp -c acl show /.:/hosts/dce_hostname/config/srvrconf
```

where *dce_hostname* is the DCE hostname of the machine to be migrated.

The output of this command should resemble this:

```
{unauthenticated -r--}
{user hosts/dce_hostname/self criI}
{group subsys/dce/dced-admin cri-}
{any_other -r--}
```

The machine context must have all the permissions as listed above. If the account does not have these permissions, you may run one of the following commands to grant the required permissions. You must first login as the cell administrator or any other account that has the permissions to modify this ACL.

If an entry does not exist for the self account:

```
dcecp -c acl modify /.:/hosts/dce_hostname/config/srvrconf \
      -add {user hosts/dce_hostname/self criI}
```

If an entry does exist but is not complete:

```
dcecp -c acl modify /.:/hosts/dce_hostname/config/srvrconf \
-change {user hosts//self criI}
```

2. Stop DCE, back up DCE data, and uninstall the old DCE product:

   ```
   dcesetup upgrade_uninstall -backdir <DCE_backup_dir>
   ```

   **Note: dcesetup** can be found on the DCE 3.1 for Solaris CD.
   This will save DCE and DFS files from:

   **/opt/dcelocal/var**
   **/opt/dcelocal/etc**
   **/krb**

3. Install Solaris 7 for SPARC.

- **Migrating DCE Clients:**

  Install DCE 3.1 for Solaris, restore backup data, and restart DCE:

  ```
  dcesetup upgrade_install -backdir <DCE_backup_dir>
  -dir<DCE_release_directory> [-mklinks <DCE_target_directory>]
  ```

  **Note: dcesetup** can be found on the DCE 3.1 for Solaris CD.

  1. Packages equivalent to the packages that you had installed for your previous level of DCE are installed. (The packages have been renamed.)

2. All files saved during the **upgrade_uninstall** process are restored.
3. DCE will be restarted. **dcesetup** invokes **start.dce** which in turn invokes **migrate.dce** to migrate all DCE configuration data to the DCE 3.1 for Solaris format.

- **Migrating DCE Security Replicas:**

  DCE security replica servers can be migrated using the steps documented in *Migrating DCE Clients* on page 46. Before enabling DCE 3.1 for Solaris function on your master security server, migrate all security replicas in your cell. When planning your migration, keep the following limitations in mind:

  1. If security replicas are migrated prior to the migration of the master security server, they will run with only your previous level of DCE for Solaris function enabled. When the DCE 3.1 for Solaris function is enabled on the security master using the **dcecp** command (given in Step 3 on page 48 under *Migrating the DCE Security Master*), these security replica servers will also enable the DCE 3.1 for Solaris function.
  2. If the DCE 3.1 for Solaris function is enabled on the master security server prior to the migration of all security replicas, any replicas which are running your previous level of DCE will be shut down. These security replicas cannot support DCE 3.1 for Solaris function.

- **Migrating DCE CDS Servers:**

  1. Ensure that all CDS master directory replicas located on this machine are replicated on at least one other CDS server machine in the cell. If you wish to support updates to these CDS directories during the migration process, move these master directory replicas to another CDS server.
  2. Perform the tasks described in *Migrating DCE Clients*, on page 46.

- **Migrating the DCE Security Master:**

  1. To minimize the impact to ongoing cell operations, ensure that at least one security server replica is running before you commence. This will support continuing security server **query** operations, though **update** operations will not be supported during the time the master security server is down.

     If the machine which is your security master server is also a CDS server, ensure that all CDS master directory replicas located on this machine are replicated on at least one other CDS server machine in the cell. If you wish to support updates to these CDS directories during the migration process, move these master directory replicas to another CDS server.
  2. Perform the tasks described in *Migrating DCE Clients*, on page 46. .

     At this point in the migration process, all your previous level of DCE for Solaris functions remain operable, but DCE 3.1 for Solaris functions are not yet enabled.

3. Enable DCE 3.1 for Solaris function, by executing the following command:

   ```
   /usr/bin/dcecp -c registry modify -version {secd.dce.1.2.2}
   ```

   It is recommended that you issue this command only after all security replica servers in your cell have been migrated to DCE 3.1 for Solaris.

   **Note:** Any security replica servers in your cell that cannot run at the OSF DCE 1.2.2 version will be shutdown after entering this command, and will be unable to be restarted. DCE 2.2 for AIX®, DCE 3.1 for AIX and Solaris and DCE 2.2 for Microsoft Windows NT™ can all run at this version.

4. After you have migrated the master security server, you need to validate any intercell accounts that exist in the DCE registry. This can be done by using the following command while logged in as the cell administrator:

   ```
   /usr/bin/dcecp -c account modify krbtgt/cell_name -change {acctvalid
   yes}
   ```

   where *cell_name* is the name of the foreign cell. If you do not validate these accounts, intercell access from some OSF 1.2.2 clients to the foreign cell might fail.

- **Migrating DTS Servers**

  To correctly migrate DTS servers, follow the tasks described in *Migrating DCE Clients* on page 46. Note that any time providers in use on the system will not be recognized by the migration tool. To continue using a time provider, it might be necessary to manually reconfigure it after the migration is completed.

- **Migrating Password Strength Servers**

  The following information will be useful when migrating a machine with a configured Password Strength server.

  The Password Strength server shipped with DCE 3.1 is an enhanced Password Strength server. This new Password Strength server will overwrite the pwd_strengthd in **/opt/dcelocal/bin**. The previous version of pwd_strengthd that was in **/opt/dcelocal/bin** will be saved as **/opt/dce.save/pwd_strengthd**.

  – **DCE-provided Password Strength server**

    If you were using the Password Strength server shipped with a previous version of DCE (pwd_strengthd), unless additional manual migration steps are performed, the new Password Strength server will support only the level of function shipped in the previous release. In order to operate in enhanced mode, the new Password Strength server requires additional configuration steps that were not performed in previous releases of DCE. These additional steps cannot be performed by the migration process

because cell administrator authority is required to perform the steps. To enable the new features, do one of the following:

- Update your existing Password Strength server configuration.
  1. Log in as the cell administrator.
  2. Run

     ```
     /opt/dcelocal/bin/migrate_pwd_strengthd
     ```
- Reconfigure the Password Strength server.

  On the Password Strength server machine:
  1. Run

     ```
     unconfig.dce pw_strength_svr
     ```
  2. Run

     ```
     config.dce pw_strength_svr
     ```

See *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components* for information on how to enable the Enhanced Password Strength server rules for DCE users' passwords.

– **Customized Password Strength servers**

If you were using a customized version of pwd_strengthd in **/opt/dcelocal/bin**, you can find your customized program in the saved location indicated previously. If you want to continue using your customized version of pwd_strengthd, move your customized pwd_strengthd to a new location then reconfigure it. If you do this, installing future versions of DCE will not overwrite your customized pwd_strengthd. Alternately, you can replace the installed pwd_strengthd with the saved pwd_strengthd if your customized Password Strength server meets the following conditions:

- The same principal name as the previous version of DCE was used (pwd_strength).
- The program can accept additional parameters on the command line without error. (The parameters ″-s pwd_strength″ will be added to the pwd_strengthd command line during migration.)

**Note:** The rules that you have defined will not be enforced until you restart DCE after replacing the installed Password Strength server.

If you wish to reconfigure your customized Password Strength server, perform the following steps on the Password Strength server machine:

1. Copy your customized password strength server from **/opt/dce.save/pwd_strengthd** to a location other than **/opt/dcelocal/bin**.
2. Unconfigure your password strength server using the following command:

   ```
   unconfig.dce -pwdstr_principal pwd_strength pw_strength_svr
   ```

3. Reconfigure your password strength server using the following command:

   ```
   config.dce -pwdstr_principal <principal name> \
   -pwdstr_cmd <fully qualified exe> \
   -pwdstr_arg <command line args> \
   pw_strength_svr
   ```

   where <principal name> is the principal name that your Password Strength server uses. The default for the previous release was pwd_strength.

   **Notes:**

   a. If you apply more than one command line argument to the **-pwdstr_arg** option, the arguments must be enclosed by double quotation marks (″). For example:

      ```
      -pwdstr_arg "-v -d"
      ```

   b. If you want to specify a password strength principal other than pwd_strengthd, specify both the**-pwdstr_principal** option and the **-pwdstr_arg -server_princ** option. For example:

      ```
      -pwdstr_principal pwd_server
      -pwdstr_arg "-server_princ pwd_server"
      ```

## Uninstalling DCE 3.1

Before uninstalling DCE for Solaris, Version 3.1, you must unconfigure your machine. See “Unconfiguring DCE Components” on page 72 for information about unconfiguration.

Once you have unconfigured DCE 3.1 for Solaris, use **dcesetup** to uninstall it:

```
dcesetup uninstall {-all | -component {appdev | cdsserver | client | secserver \
| sysmgmt | priv | msgs | docs}
[-nomsgs] [-msg_langs {<en_US> | <it> | <es> | <ja> | ja_JP.PCK>| <ko> | \
 <zh> | <zh.GBK>}]
[-nodocs] [-doc_langs {<en_US> | <it> |  <ko> | <zh>}]
```

You can also use the **pkgrm** command to uninstall DCE. The **IDCEclnt** package must be the last package you remove.

```
pkgrm IDCEcdss IDCEsecs IDCEclnt
or
pkgrm IDCEcdss
pkgrm IDCEsecs
pkgrm IDCEclnt
```

See “Installable Packages” on page 43 for a complete list of package names.

## Suggested Reading

For information on configuring a DCE cell, see the “Configuring DCE” on page 55 and the **config.dce** command in the *IBM DCE Version 3.1 for Solaris: Administration Commands Reference* .

For information about unconfiguring individual DCE components, see the **unconfig.dce** command in the *IBM DCE Version 3.1 for Solaris: Administration Commands Reference*.

# Part 3. Configuring, Starting, and Stopping DCE 3.1 for Solaris

# Chapter 4. Configuring DCE 3.1 for Solaris Servers and Clients

## Configuring DCE

The following sections describe creating and configuring a DCE cell:
- "Overview of Configuration"
- "Initial Cell Configuration" on page 61
- "Further Cell Configuration" on page 65
- "Unconfiguring DCE Components" on page 72

These sections include server components and client components for the following DCE services: security service, Cell Directory Service (CDS), Distributed Time Service (DTS), Remote Procedure Call (RPC), and Global Directory Agent (GDA).

For information on setting up the intercell environment, managing intercell naming, and administering a multicell environment, see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components*.

## Overview of Configuration

The configuration of a DCE cell occurs in two phases. During the first phase, or *initial cell configuration*, certain tasks must be performed to initialize the cell. During the second phase, generic tasks can be performed to configure (or reconfigure) additional features into the cell.

A DCE cell requires the following components:
- One security server
- One CDS server

It is recommended that there also be at least one DTS server (although three or more DTS servers are preferred for accuracy of time synchronization).

The security and CDS servers must be configured to initialize any cell. After the cell is up and running, you generally will not have to repeat any of these configuration tasks.

Additional components that can be configured into a cell are the following:

- DCE clients
- Secondary CDS servers
- Replica security servers
- Audit Services
- Global Directory Agents (GDAs)
- DTS Services
- Simple Network Management Protocol (SNMP)
- Event Management Service (EMS)
- Password Strength Server
- Security Integration (Pluggable Authentication Module [PAM] and Name Service Switch [NSS])
- Identity Mapping Service (IDMS)
- Name Service Interface Daemon (NSID)
- DCE Web Secure

The configuration of these additional components is a task you can perform at any point throughout the lifetime of the cell after initialization.

Keep the following items in mind when you are configuring a cell:

- For better performance and reliability install the master security server and the initial CDS server on different machines.
- Clients can be configured in one of three ways:

  **Split Configuration**
  This type of configuration is used when the DCE cell administrator is unlikely to have root user access to every machine in the cell. It is comprised of two distinct sets of operations:

  **admin** This type of configuration updates the namespace and security registry with information about the new client. The cell administrator must run the **config.dce** command from a machine within the existing cell. It cannot be run from the new client machine. The cell administrator does not need root user authority to run the admin portion of configuration.

  **local** This type of configuration creates the necessary files on the local machine and starts the daemons for the new client. The admin part of config.dce must have been run first or the local configuration will fail when trying to contact the cell. The user must have root authority on the machine, but does not need to have any authority in the DCE cell. The following components do not require that the admin part of **config.dce** be run first: slim client, audit, integrated login, and RPC.

**Full Configuration**
This type of configuration is the default. Full configuration includes admin configuration steps and local configuration steps. The DCE cell administrator must have root authority on the local machine being configured into the cell.

- Before configuring a machine into a cell, make sure that the machine's clock is within five minutes of the cell's master security server's clock. If the machine's clock is skewed more than five minutes, authentication errors might result, and configuration might fail. If you have already configured at least one DTS server in the cell, you can use the **-sync_clocks** flag to perform the synchronization for you automatically.
- If you want to reconfigure a particular component (or an entire machine) with new parameters, you must unconfigure it to remove the existing configuration before setting up the new configuration.
- To enable intercell communication that uses GDA, you must also register the cell's name into a global directory, such as the Domain Name System (DNS). For information on the intercell environment, see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components*.

DCE 3.1 for Solaris provides the following commands to perform configuration tasks at the command line:

**chpesite**
Updates the **pe_site** file, which contains the addresses of the security servers that you use.

**clean_up.dce**
Cleans up recreatable database files, cache files, and credential files. This commands is intended to be used if problems are encountered when trying to start DCE.

**config.dce**
Configures and starts DCE components. This command provides for a *split configuration of clients*. Administrative configuration and local configuration can be performed separately. See "Further Cell Configuration" on page 65 for more information.

**kerberos.dce**
Creates the host principals, FTP principals, and key table entries that are used to support kerberos interoperability with DCE.

**migrate.dce**
Migrates DCE configuration data from previous releases for use with the current release. There is no need to reconfigure when installing a new release of DCE.

**mkdceweb**
Configures DCE Web Secure into a Netscape FastTrack 3.01 or Enterprise 3.61 Web server.

**mkreg.dce**
Adds information about a DCE cell into the DOMAIN namespace.

**rmdceweb**
Unconfigures DCE Web Secure from a Netscape FastTrack 3.01 or Enterprise 3.61 Web server.

**rmreg.dce**
Removes information about a DCE cell from the DOMAIN namespace (DNS).

**show.cfg**
Displays the local host's DCE or DFS configuration. The **dce** and **dfs** options allow display of only DCE or DFS information

**start.dce**
Starts the configured DCE components. This command makes sure that all components are started in the correct order.

**stop.dce**
Stops the configured DCE components. This command makes sure that all components are stopped in the correct order.

**unconfig.dce**
Removes configurations of DCE components. This command provides for a *split unconfiguration*, with which administrative configuration and local configuration can be performed separately. See "Further Cell Configuration" on page 65 for more information.

For detailed information on these commands, refer to the *IBM DCE Version 3.1 for Solaris: Administration Commands Reference*.

## User-Supplied Commands

The DCE 3.1 for Solaris config/unconfig/start/stop code now provides support for user-supplied commands. User-supplied commands can be executed before or after, or before and after configuration, unconfiguration, start and stop of DCE. The intent of this support is to allow you to run your own commands without having to modify the scripts that are shipped with the DCE product. When a future release of this product is installed, your user-supplied commands will automatically run with the new release.

Perform the following:

Write your command to do what you need. When executed, the configuration commands,(**config.dce**, **unconfig.dce**, **start.dce**, and **stop.dce**), set the environment variable, **"callers_cmd_line"**, (including all the parameters with the exception of the cell administrator's password) to the command line. For example, when configuring DCE, if the command executed is:

```
"config.dce -cell_name mycellname -admin_pwd -dce-
sec_srv cds_srv"
```

The **callers_cmd_line** environment variable is set to:

```
"-cell_name mycellname -admin_pwd <****>
sec_srv cds_srv"
```

This environment variable might be useful to your command script.

Create the file **/opt/dcelocal/tcl/user_cmd.tcl**. This file should contain the appropriate subset of the following entries:

For DCE:

- **set pre_config_dce** — the full path to your pre-DCE configuration command and any arguments
- **set pre_config_dce_fail_on_error** — **$TRUE** or **$FALSE** to indicate whether the **config.dce** should fail (or not) if your command fails.
- **set post_config_dce** — the full path to your post-DCE configuration command and any arguments
- **set post_config_dce_fail_on_error** — **$TRUE** or **$FALSE** to indicate whether the **config.dce** should fail (or not) if your command fails.
- **set pre_unconfig_dce** — the full path to your pre-DCE unconfiguration command and any arguments
- **set pre_unconfig_dce_fail_on_error** — **$TRUE** or **$FALSE** to indicate whether the **unconfig.dce** should fail (or not) if your command fails.
- **set post_unconfig_dce** — the full path to your post-DCE unconfiguration command and any arguments
- **set post_unconfig_dce_fail_on_error** — **$TRUE** or **$FALSE** to indicate whether the **unconfig.dce** should fail (or not) if your command fails.
- **set pre_start_dce** — the full path to your pre-DCE start command and any arguments
- **set pre_start_dce_fail_on_error** — **$TRUE** or **$FALSE** to indicate whether the **start.dce** should fail (or not) if your command fails.
- **set post_start_dce** — the full path to your post-DCE start command and any arguments

- **set post_start_dce_fail_on_error** — **$TRUE** or **$FALSE** to indicate whether the **start.dce** should fail (or not) if your command fails.
- **set pre_stop_dce** — the full path to your pre-DCE stop command and any arguments
- **set pre_stop_dce_fail_on_error** — **$TRUE** or **$FALSE** to indicate whether the **stop.dce** should fail (or not) if your command fails.
- **set post_dce** — the full path to your post-DCE stop command and any arguments
- **set post_stop_dce_fail_on_error** — **$TRUE** or **$FALSE** to indicate whether the **stop.dce** should fail (or not) if your command fails.

**Note:** Use # to include a comment on its own line. Use ;# to include a comment on a line of code.

The configuration, unconfiguration, start, and stop codes look for the **/opt/dcelocal/tcl/user_cmd.tcl** file and the following variable names:

**pre_config_dce**
**post_config_dce**
**pre_unconfig_dce**
**post_unconfig_dce**
**pre_start_dce**
**post_start_dce**
**pre_stop_dce**
**post_stop_dce**

**pre_config_dce_fail_on_error**
**post_config_dce_fail_on_error**
**pre_unconfig_dce_fail_on_error**
**post_unconfig_dce_fail_on_error**
**pre_start_dce_fail_on_error**
**post_start_dce_fail_on_error**
**pre_stop_dce_fail_on_error**
**post_stop_dce_fail_on_error**

Examples:

```
#Set some environment variables before configuring DCE
#config.dce should fail if /usr/bin/set_env_vars fails
set pre_config_dce "/usr/bin/set_env_vars"
set pre_config_dce_fail_on_error $TRUE

#The following command runs the App XYZ config command
#App XYZ must be configured after DCE
#config.dce will NOT fail if /usr/bin/APP_XYZ_config fails
set post_config_dce "/usr/bin/APP_XYZ_config -arg1 arg1_value -arg2 arg2_value"

#The following command runs the App XYZ start command
#App XYZ must start after DCE
#start.dce wil NOT fail if /usr/bin/APP_XYZ_start fails
set post_start_dce "/usr/bin/APP_XYZ_start"
set post_start_dce_fail_on_error $FALSE
```

```
#Stop App ABC before stopping DCE
set pre_stop_dce "/usr/bin/APP_ABC_stop"
#stop.dce will fail if /usr/bin/APP_ABC_stop fails
set pre_stop_dce_fail_on_error $TRUE
```

### Environment Variables

Environment variables are variables that are used by DCE that customers can set themselves. See the *IBM DCE Version 3.1 for Solaris: Administration Guide—Introduction* for more information about DCE environment variables.

## Initial Cell Configuration

To initialize a cell, you must perform these basic tasks in order:

1. Configure the master security server machine. See "Configuring the Master Security Server" on page 62.
2. Configure the initial CDS server machine. See "Configuring the Initial CDS Server" on page 62.
3. Configure a CDS client on the master security server. See "Configuring a CDS Client on the Master Security Server" on page 65.

In the procedures that follow, ensure that the *dce_hostname* of each machine is unique within the cell. The *dce_hostname* is the name that is listed in the hosts directory (**hosts/***dce_hostname*) in the namespace. The **config.dce** command allows you to assign a *dce_hostname* independent of a machine's host name on the network. By default the host name of the machine is used.

**Attention:** If you attempt to configure two machines that have the same *dce_hostname*, you will have to unconfigure and reconfigure DCE on both machines. If one of these machines is either the security server or the initial CDS server, you will have to unconfigure and reconfigure DCE on *every* machine in the cell.

The following sections provide detailed procedures for performing these initial configuration tasks.

### Configuring Servers

This section discusses the following:

- "Configuring the Master Security Server" on page 62
- "Configuring the Initial CDS Server" on page 62

### Configuring the Master Security Server

To configure the master security server for a cell, perform the following steps on the machine that is designated as the master security server:

To configure the master security server for a cell, on the master security server machine, log in as root and type the following at a command line:

```
config.dce -cell_name <cell_name> [-sec_server_name <security_server>]
[-cell_admin <cell_admin_id>] [-admin_pwd <admin_password>]
[-min_princ_id <min_principal_id>] [-min_group_id <min_group_id>]
[-min_org_id <min_org_id>] [-max_unix_id <max_UNIX_id>]
[-no_pesite_update] [-pesite_update_time <update_time>]
[-autostart yes | no] [-clean_autostart yes | no] [-protocol tcp udp]
[-certificate_based_login yes | no ] [-kdc_profile <kdc_profile>]
[-kdc_ini_file <kdc_ini_file> ] [-kdc_passphrase <kdc_passphrase>]
[-group_rsp_path <filename>] [-rsp_file <filename>]
sec_srv
```

At this point, **dced** (RPC and a security client) and the master security server are configured on the machine. If you configure the initial CDS server on another machine, you must configure a CDS client on the master security server machine before you configure anything else. See “Configuring a CDS Client on the Master Security Server” on page 65. You can return to this machine later to configure other DCE components.

### Configuring the Initial CDS Server

There can be only one *initial* CDS server for each cell. To configure the initial CDS server for a cell, on the initial CDS server machine, log in as root and type the following at a command line:

```
config.dce [-cell_name <cell_name>] [-cell_admin <cell_admin_id>]
[-admin_pwd <admin_password>] [-sec_master <master_security_server>]
[-autostart yes | no] [-clean_autostart yes | no] [-protocol tcp udp]
[-group_rsp_path <filename>] [-rsp_file <filename>]
cds_srv
```

At this point, **dced** (RPC and a security client), the initial CDS server, and a CDS clerk are configured on the machine. (If this machine is the master security server, only the initial CDS server and a CDS clerk are actually configured in this section.) You can return to this machine later to configure DTS.

Note that a clearinghouse is automatically created when you configure a CDS server. Although it is possible to define multiple clearinghouses for a CDS server, you should have only one per CDS server during normal operation. If you are moving a clearinghouse from one CDS server to another, however, you can temporarily define a second clearinghouse on the original server. See

the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components* for more information on moving a clearinghouse.

## Configuring Clients

This section discusses the following:



Typically, you need to configure many clients into a DCE cell. Configuring clients entails two distinct sets of operations:

- Tasks that require *cell administrator* authority within the DCE cell
- Tasks that require *root user* authority on the machine that is to be configured as a DCE client.

These tasks are separated into a *split configuration of clients* because a DCE cell administrator is unlikely to have root user access to every machine in a cell.

### Configuring DCE Clients

The DCE clients can be configured in one of three ways: full, admin, or local.

Split client configuration for security clients (**sec_cl**) and CDS clients (**cds_cl**) is a two-part process. (The cell administrator might not have root access to the client machines, or the root user might not have cell administrator access.) The two parts are the following:

- The cell administrator runs the **admin** portion from any machine in the cell to update the CDS namespace and security registry.
- The root user of the client machine runs the **local** portion to create necessary files and to start client daemons for all client components.

### Admin Client Configuration

To do the **admin** portion of configuring a DCE client, log in to any machine in the cell as cell administrator and type the following at a command line:

```
config.dce -config_type admin -host_id <machine identifier>
[dce_hostname <dce_hostname> [-cell_admin <cell_admin_id>
[-admin_pwd <admin_password>] [-lan_profile <profile>]
[-protocol tcp udp] [-group_rsp_path <filename>]
[-rsp_file <filename>]
sec_cl | cds_cl | dts_cl | all_cl
```

At this point, the namespace entries and security registry database have been updated. It is now necessary to run the **local** portion of configuration to complete the process.

### Local Client Configuration

To do the **local** portion of configuring a DCE client (after the **admin** portion is completed, when appropriate), log in to the client machine as root and type the following at a command line:

```
config.dce -config_type local
[-cell_name <cell_name>] [-dce_hostname <dce_hostname>]
[-sec_master <master_security_server>] [-cds_server <cds_server>]
[-no_pesite_update] [-pesite_update_time <update_time>]
[-time_server <server id>] [-sync_clocks yes | no] [-autostart yes | no]
[-clean_autostart yes | no] [-protocol tcp udp] [-proxy]
[-group_rsp_path <filename>] [-rsp_file <filename>]
[-num_dce_unixd <number>] [cache_lifetime <minutes>]
[-cds_replica_list <list_of_cds_servers>]
sec_cl | cds_cl | dts_cl | all_cl
```

At this point, the selected clients are configured on the machine.

### Full Client Configuration

If you are both the *cell administrator* and the *root user* of a machine currently being configured as a client, you can perform a **full client** configuration, which incorporates both the **admin** and **local** portions of configuration.

To perform the **full** configuring of a DCE client, log in on your machine as root and type the following at a command line::

```
config.dce -config_type full
[-cell_name <cell_name>] [-dce_hostname <dce_hostname>]
[-admin_pwd <admin_password>] [-cell_admin <cell_admin_id>]
[-sec_master <master_security_server>] [-cds_server <cds_server>]
[-lan_profile <profile>] [-pesite_update_time <update_time>]
[-no_pesite_update]  [-time_server <server id>] [-sync_clocks yes | no]
[-autostart yes | no] [-clean_autostart yes | no] [-protocol tcp udp]
[-proxy] [-group_rsp_path <filename>] [-rsp_file <filename>]
[-num_dce_unixd <number>] [cache_lifetime <minutes>]
[-cds_replica_list <list_of_cds_servers>]
sec_cl | cds_cl | dts_cl | all_cl
```

At this point, the selected clients are configured on the machine.

### Slim Client Configuration

Ensure that the **CELL name** field is filled in with the appropriate values.

**Notes:**

1. The cell administrator's password is not needed when configuring a slim client.
2. Only a DFS client and Security Integration can be configured with a slim client.
3. There are no admin configuration steps to perform prior to configuring a slim client.

To configure a DCE slim client on the client machine log in as root and type the following at a command line:

```
config.dce -cell_name <cell_name>
[-dce_hostname <dce_hostname>] [-sec_master <master_security_server>]
[-cds_server <cds_server>] [-time_server <server id>] [-sync_clocks yes | no]
[-autostart yes | no] [-clean_autostart yes | no] [-protocol tcp udp]
[-group_rsp_path <filename>] [-rsp_file <filename>]
slim_cl
```

At this point, the selected clients are configured on the machine.

### Configuring a CDS Client on the Master Security Server

If you configured the master security server and the initial CDS server on the same machine, you can skip this section because a CDS client was configured when you configured the initial CDS server.

Otherwise, to configure a CDS client on the master security server, type the following at a command line on the machine that is the master security server:

```
config.dce [-cell_admin <cell_admin>] cds_cl
```

At this point, a CDS client is configured on the machine.

## Further Cell Configuration

After cell initialization is completed, you might have to perform additional configuration tasks on an ongoing basis as changes are made to the cell. For example, you might want a new machine to be added to the cell as a client. Or you might decide to configure a secondary CDS server to provide faster or more reliable access to the namespace.

The following sections provide detailed procedures for performing additional configuration tasks.

## Configuring DTS Servers

To configure DTS local or global servers, log in as root and type the following at a command line on each machine that is designated as a DTS server:

```
config.dce [-courier_role (courier | noncourier | backup)]
[-cell_name <cell_name>] [-cell_admin <cell_admin_id>]
[-admin_pwd <admin_password>] [-sec_master <master_security_server>]
[-cds_server <cds_server>] [-lan_profile <profile>]
[-time_server <server id>] [-sync_clocks yes | no]
[-autostart yes | no] [-clean_autostart yes | no] [-protocol tcp udp]
[-group_rsp_path <filename>] [-rsp_file <filename>]
dts_local | dts_global
```

At this point, a DTS server is configured on the machine, along with **dced** (RPC and a security client) and a CDS client which were configured as part of DCE client configuration.

## Configuring a DTS Client

To configure a DTS client, log in as root and type the following at a command line of those machines:

```
config.dce [-cell_admin <cell_admin_id>] dts_cl
```

## Configuring Secondary CDS Servers

After you have configured an initial CDS server, you might want to configure one or more *secondary* CDS servers to provide faster or more reliable access to the namespace.

A Secondary CDS Server allows administrators to create replicas of CDS Directories for backup and availability purposes. When you configure a Secondary CDS Server, a replica of the root directory and its contents is automatically created.

The only child directory below the root that is automatically replicated into the new Secondary CDS Server is the **/.:/subsys/dce/sec** directory. This directory is replicated because it contains the binding information to locate the master security server. This action increases accessibility to the security server even when the initial CDS Server is unavailable. See the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components* for information about CDS replicas and how to create them.

To configure a secondary CDS server, on each machine that is designated as a secondary CDS server, log in as root and type the following at a command line::

```
config.dce [-cell_name <cell_name>] [-cell_admin <cell_admin_id>]
[-admin_pwd<admin_password>] [-sec_master <master_security_server>]
[-cds_server <cds_server>] [-lan_profile <profile>]
```

```
[clr_house <server id>] [-autostart yes | no] [-clean_autostart yes | no]
[-protocol tcp udp] [-time_server <server id>] [-sync_clocks yes | no]
[-group_rsp_path <filename>] [-rsp_file <filename>]
cds_second
```

At this point, **dced** (RPC and a security client) a secondary CDS server, and a CDS client are configured on the machine. When you configure a secondary CDS server, only the **root** and the **/.:/subsys/dce/sec** directories are replicated. See the *IBM DCE Version 3.1 for Solaris: Administration Guide* for information on replicating other directories.

## Configuring Security Replica Servers

A security replica server is a read-only copy of the master security server. Advantages of using a security replica server include easing the load on the master security server and preserving the cell in case the master security server becomes disabled.

To configure a security replica server, on each security replica server machine, log in as root and type the following at a command line:

```
config.dce [-sec_server_name <security_server>] [-cell_name <cell_name>]
[-cell_admin <cell_admin_id>] [-admin_pwd<admin_password>]
[-sec_master <master_security_server>] [-cds_server <cds_server>]
[-autostart yes | no] [-clean_autostart yes | no] [-protocol tcp udp]
[-time_server <server id>] [-sync_clocks yes | no]
[-certificate_based_login yes | no] [-kdc_profile <kdc_profile>]
[-kdc_ini_file <kdc_ini_file>] [-kdc_passphrase <kdc_passphrase>]
[-group_rsp_path <filename>] [-rsp_file <filename>]
sec_rsp
```

At this point, **dced** (RPC and a security client), a security replica, and a CDS client are configured on the machine.

## Configuring the Global Directory Agent

The Global Directory Agent (GDA) allows intercell communication by locating a foreign cell which has been registered into the Domain Naming System (DNS) global directory service. Only one GDA is required to be configured within the cell to allow intercell communication, but more than one GDA can be configured to increase availability.

To configure the GDA on a machine, log in as root and type the following at a command line:

```
config.dce [-cell_name <cell_name>] [-admin_pwd<admin_password>]
[-cell_admin <cell_admin_id>] [-sec_master <master_security_server>]
[-cds_server <cds_server>] [-lan_profile <profile>]
[-ldap_server <ldap_server | ldap_server:port_number>]
```

```
[-time_server <server id>] [-sync_clocks yes | no]
[-autostart yes | no] [-clean_autostart yes | no] [-protocol tcp udp]
[-group_rsp_path <filename>] [-rsp_file <filename>]
gda_srv
```

At this point, **dced** (RPC and a security client), a CDS client, and the GDA are configured on the machine. To enable intercell communication, see the information on the intercell environment in the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components*. Also, see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components* for information on registering a cell globally.

## Configuring EMS Servers

To configure an EMS server, log in as root and type the following at a command line:

```
config.dce [-cell_name <cell_name>] [-admin_pwd<admin_password>]
[-cell_admin <cell_admin_id>] [-sec_master <master_security_server>]
[-cds_server <cds_server>] [-lan_profile <profile>]
[-time_server <server id>] [-sync_clocks yes | no]
[-autostart yes | no] [-clean_autostart yes | no] [-protocol tcp udp]
[-group_rsp_path <filename>] [-rsp_file <filename>]
ems_srv
```

At this point, an EMS server, **dced** (RPC and a security client) and a CDS client are configured on this machine.

## Configuring SNMP Servers

To configure an SNMP server, log in as root and type the following at a command line:

```
config.dce
[-autostart yes | no] [-clean_autostart yes | no]
snmp_srv
```

## Configuring DCE 3.1 for Solaris Security Integration

To configure a system for security integration operations, log in as root and type the following at a command line:

```
config.dce pam
```

At this point, a PAM server, **dced** (RPC and a security client) and a CDS client are configured on the machine. To set up the machine to use DCE security integrated login see the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components* for complete details.

## Configuring Audit Servers

**Note:** To allow the server to use auditing, you must have the environment variable set DCEAUDITON=1. The easiest way to accomplish this is to configure audit and then stop and restart the servers, making sure the dceauditon environment variable is set before you start.

To configure an Audit server, log in as root and type the following at a command line:

```
config.dce [-cell_name <cell_name>]
[-sec_master <master_security_server>] [-cds_server <cds_server>]
[-lan_profile <profile>][-autostart yes | no] [-clean_autostart yes | no]
[-protocol tcp udp][-time_server <server id>] [-sync_clocks yes | no]
[-group_rsp_path <filename>] [-rsp_file <filename>]
audit
```

At this point, an Audit server, **dced** (RPC and a security client) and a CDS client are configured on this machine.

## Configuring Password Strength Servers

To configure a Password Strength server on a machine, log in as root and type the following at a command line:

```
config.dce [-cell_name <cell_name>] [-cell_admin <cell_admin_id>]
[-admin_pwd <admin_password>] [-sec_master <master_security_server>]
[-cds_server <cds_server>][-lan_profile <profile>]
[pwdstr_arg <command line args>] [-pwdstr_cmd <server_name>]
[pwdstr_principal <password strength principal id>]
[-autostart yes | no] [-clean_autostart yes | no] [-protocol tcp udp]
[-time_server <server id>] [-sync_clocks yes | no]
[-group_rsp_path <filename>] [-rsp_file <filename>]
pw_strength_srv
```

**Notes:**

1. If you apply more than one command line argument to the **-pwdstr_arg** option, the arguments must be enclosed by double quotation marks (″). For example:

   ```
   -pwdstr_arg "-v -d"
   ```

2. If you want to specify a password strength principal other than pwd_strengthd, specify both the**-pwdstr_principal** option and the **-pwdstr_arg -server_princ** option. For example:

   ```
   -pwdstr_principal pwd_server
   -pwdstr_arg "-server_princ pwd_server"
   ```

At this point, a password strength server, **dced** (RPC and a security client) and a CDS client are configured on the machine.

### Configuring the Name Service Interface Daemon (NSID)

To configure the NSID on a machine, log in as root and type:

```
config.dce [-cell_name <cell_name>] [-cell_admin <cell_admin_id>]
[-sec_master <master_security_server>] [-cds_server <cds_server> ]
[-lan_profile <profile>][-time_server <server id>] [-rsp_file <filename>]
[-nsid_pwd <nsid_password>] [-wrap_audit_trail yes | no]
nsid
```

At this point, NSID, **dced** (RPC and a security client) and a CDS client are configured on the machine.

### Configuring an Identity Mapping Server

The Identity Mapping server must be configured on the same machine as either the security master server or a security replica server. At the prompt type:

```
config.dce idms_srv
```

When prompted, type the cell administrator's password.

At this point, an Identity Mapping server, security server (master or replica), **dced** (RPC and a security client), and CDS client are configured on the machine.

## Configuring DCE Web Secure for Solaris

DCE Web Secure must be installed and configured on a workstation that has a Netscape Enterprise 3.61 or a FastTrack 3.01 Web server and a DCE client.

To configure DCE Web Secure for Solaris from the command line, type in the following:

```
mkdceweb -n <netscape_dir>  -s <netscape_id>  -i <user_id> -t secure
```

- The <netscape_dir> is the root directory where your Netscape server product is installed. The default Netscape home directory is */usr/ns-home*.
- The <netscape_id> identifies the name of the Web server. This value comes from the **Server Identifier** field specified by the administrator through the Netscape Administration Server when the server was installed.
- The <user_id> is the operating system user account name under which the Netscape server should run. The <user_id> cannot be *nobody*, and it is recommended that the Web server not be run as *root*.

## Verifying Configuration of DCE Web Secure

The best way to verify whether or not your configuration was successful, besides the absence of error messages during configuration, is to use the features of DCE Web Secure through your Web browser.

### Accessing a CGI using DCE Credentials

With DCE Web Secure installed and configured, you can provide DCE credentials to Common Gateway Interface (CGI) programs. This functionality can be especially useful if you want to run a shell script or tcl script that needs DCE credentials from your Web browser.

As an example, place the following shell script in a file called testcgi.sh in a directory from which the Web server will allow CGIs to be run. Also, ensure the shell script has execute permission for the operating system userid under which the Web server is running.

```
#!/bin/sh
 # testcgi.sh Test CGI program to show DCE credentials.
 echo "Content-type: text/html"
 echo ""
 echo "<html>"
 echo "<head>"
 echo "<title>Test CGI program to show DCE credentials</title>"
 echo "</head>"
 echo "<body>"
 echo "<h1>Test CGI program to show DCE credentials</h1>"
 echo "<p><h3>CGI is running under the following DCE credentials:</h3>"
 echo "<pre>"
 klist | grep "Global Principal"
 echo "</pre>"
 echo "</body>"
 echo "</html>"
```

When run, the CGI will show you which DCE credentials you are using. For example, if you stored the file in **/usr/opt/dcelocal/web/admin/cgi-bin/testcgi.sh** and you setup your Web server to allow CGI programs to run from **/usr/opt/dcelocal/web/admin/cgi-bin**, you can run this CGI from the following URL:

```
http://<server-name>/dceweb/cgi_bin/testcgi.sh
```

To ensure that the CGI runs with DCE credentials, check that unauthenticated access is not turned on for the **/usr/opt/dcelocal/web/admin/cgi-bin** path in your Web server configuration.

## Unconfiguring DCE Components

Occasionally, certain situations require that you unconfigure (or remove configuration and database files for) a particular DCE component from a machine. For example, if you want to reconfigure a particular component with new parameters, you must unconfigure it to remove the existing configuration before setting up the new configuration. Or, if configuration of a component failed and it is only partially configured, you must remove the partial configuration before attempting configuration again.

Other situations require that you unconfigure an entire machine (that is, unconfigure all DCE components from the machine). For example, if you want to transfer a machine from one cell to another, you must remove the configurations for the old cell from the machine before setting up the configurations for the new cell.

In rare cases, you might want to unconfigure an entire cell. If you unconfigure a cell, you should also unregister its name from the global namespace.

**Attention:** After you unconfigure a secondary CDS server (**unconfig.dce cds_second**), you must wait 24 hours before you reconfigure a secondary CDS server with the same name.

The following section provides more information on unconfiguring DCE components.

### Considerations Before Unconfiguring

You should exercise caution in unconfiguring DCE components, especially if you are removing components which perform services that are required by other components. Unconfiguring a component will partially or completely disable other components which are dependent upon it.

**Attention:** In the event that you must reconfigure a cell and you are running DFS in your environment, refer to your DFS documentation supporting IBM DCE 3.1 for considerations before reconfiguring a cell.

There are special cases which you should take into consideration when unconfiguring DCE components:

- The master security server and the CDS server that contain the master replica of the **/.:** directory are the basis of any cell. If you unconfigure one or both of these servers, you have to unconfigure and rebuild your entire cell.
- To unconfigure a CDS server that has a master replica of any directory, you must use the **local** option.

See the *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components* for more information on changing the location of a directory's master replica.

When you unconfigure DCE components on a machine, two types of operations are performed:

- Local operations (updating configuration files and stopping daemons)
- Administrative operations (updating the security registry, the CDS namespace).

Just as configuration is separated into **admin** and **local** portions, so is most of unconfiguration. The exceptions are the master security server (**sec_srv**) and any CDS server (**cds_srv** or **cds_second**) that contains a master replica of a directory in one of its clearinghouses.

When you unconfigure DCE components on a machine, if all the local operations can be undone, the machine itself is considered to be unconfigured (locally only). However, if attempts to undo administrative operations fail, the machine is not fully unconfigured from the cell; entries for the machine might still exist in the CDS namespace or registry databases. On a full unconfiguration if attempts to undo administrative operations fail, a list of the failed operations is printed so you can manually perform these operations and remove references to the machine from the namespace and registry databases. From another machine configured in your cell, you can run **admin** unconfiguration for operations that failed so that you can clean up the DCE registry database and the namespace.

Refer to the *IBM DCE Version 3.1 for Solaris: Administration Commands Reference* for complete information on the DCE commands that are referenced above.

## Split Unconfiguration

Sometimes it is beneficial to use a feature known as the *split unconfiguration of clients*, which allows the root user to perform the unconfiguration steps on the local machine while the cell administrator cleans up the rest of the cell. A **local** unconfiguration is useful in the following situations:

- If the cell for which a machine is configured is inaccessible or you do not have the password for that cell administrator's account, you need only to remove the local configuration files from the machine to reconfigure it for a new cell.
- If the configuration of a machine is so broken that it cannot reach the Security server to be authenticated to perform remote operations, you can limit unconfiguration to local items.
- If you are unconfiguring a master security server that contains the master replica of a directory, you can limit unconfiguration to local items.

- If you are unconfiguring a CDS server that contains the master replica of a directory, you can limit unconfiguration to local items.
- If you are unconfiguring a slim client, only local unconfiguration steps are necessary

The cell administrator should run the **admin** portion of unconfiguration from a machine in the cell to complete the unconfiguration process. A full client that has been locally unconfigured cannot be configured back into the cell until the admin portion of unconfiguration has been done.

## Steps for Unconfiguring DCE

To fully unconfigure one or more DCE components from a machine, as root at a command line of the machine type:

```
unconfig.dce -config_type full
[-cell_admin <cell_admin_id>] [admin_pw <admin_pw>] [-dependents]
[-force] [-pwdstr_principal <password_strength_principal_id>]components
```

**Note:** For a full unconfiguration, **components** can be any DCE components *except* sec_srv or cds_srv.

To locally unconfigure one or more DCE components from a machine, as root at a command line of the machine type:

```
unconfig.dce -config_type local
[-dependents] [-force]
[-pwdstr_principal <password_strength_principal_id>]
components
```

**Note:** For a local unconfiguration, **components** can be any DCE components.

To perform an admin unconfiguration of one or more DCE components from another machine, as root at a command line type:

```
unconfig.dce -config_type admin
[-dce_hostname <dce_hostname>]
[-cell_admin <cell_admin_id>] [admin_pw <amdin_pw>]
[host_id <machine identifier>]
[-dependents] [-force]
[-pwdstr_principal <password_strength_principal id>]components
```

**Note:** For an admin unconfiguration, **components** can be any DCE components *except* sec_srv or cds_srv.

## Unconfiguring DCE Web Secure

To unconfigure DCE Web Secure for Solaris, at a command line type:

```
rmdceweb -n <netscape_dir> -s <netscape_id> -t secure
```

- The <netscape_dir> is the root directory where your Netscape server product is installed. The default Netscape home directory is */usr/ns-home*.
- The <netscape_id> identifies the name of the Web server. This value comes from the **Server Identifier** field specified by the administrator through the Netscape Administration Server when the server was installed.

# Chapter 5. Starting and Stopping DCE 3.1 for Solaris

## Starting DCE Daemons

### Using the Command Line to Start Daemons

The **start.dce** command starts DCE daemons for configured DCE components. Before starting DCE daemons, you must be logged in as root.

To start all daemons for configured DCE components, type any of the following at the command line:

```
start.dce all
start.dce core
start.dce
```

To start specific configured components, add the component name, such as **cds_srv**, to the command:

```
start.dce cds_srv
```

**Note:** If the master security server and the initial CDS server are on different machines and both have been stopped, use the following steps to restart DCE:

**Machine 1**
(rpc, sec_cl, sec_srv, cds_cl, and any other dce components)

**Machine 2**
(rpc, sec_cl, cds_srv, cds_cl, and any other dce components)

1. Machine 1: **start.dce rpc sec_cl sec_srv**
2. Machine 2: **start.dce rpc sec_cl cds_cl cds_srv**
3. Machine 1: **start.dce all**
4. Machine 2: **start.dce all**

### Starting DCE Now and at System Restart

You can run **start.dce** now to start all configured DCE and DFS daemons. You can also run **start.dce** at system restart.

To start DCE now, at a command line type:

```
start.dce
```

To start DCE at system restart, at a command line type:

```
config.dce -autostart <yes | no>
```

## Stopping DCE Daemons

The **stop.dce** command stops DCE daemons for configured DCE components. To stop DCE daemons, you must be logged in as root.

To stop all daemons for configured DCE components, type one of the following at the command line:

```
stop.dce core
stop.dce
stop.dce all
```

To stop specific daemons for configured DCE components, add the daemon's name to the **stop.dce** command. For example, to stop the dts client, type the following at the command line:

```
stop.dce dts_cl
```

# Chapter 6. Obtaining Additional Information

This chapter describes the sources of information that can be useful when you are using DCE 3.1 for Solaris.

## Books

The DCE 3.1 for Solaris library contains a printed copy and an online version of the *IBM DCE Version 3.1 for Solaris: Quick Beginnings* and a printed copy of the *IBM DCE Version 3.1 for Solaris: Release Notes*. All other supporting product documentation is provided only in online format.

## Online Information

Extensive online documentation is shipped as part of the DCE for Solaris product.

### HTML Books

The manuals included with this product are in Hypertext Markup Language (HTML) softcopy format. The softcopy format makes it easier to share the library across your site.

Although you can use any browser that supports HTML 3.2, a Netscape browser is provided with the Operating System. For instructions on installing and using the browser, see the Solaris documentation.

The following IBM DCE books are available online:

- *IBM DCE Version 3.1 for Solaris: Quick Beginnings*
- *IBM DCE Version 3.1 for Solaris: Introduction to DCE*
- *IBM DCE Version 3.1 for Solaris: Application Development Guide—Introduction and Style Guide*
- *IBM DCE Version 3.1 for Solaris: Application Development Guide—Core Components*
- *IBM DCE Version 3.1 for Solaris: Application Development Guide—Directory Services*
- *IBM DCE Version 3.1 for Solaris: Application Development Reference*
- *IBM DCE Version 3.1 for Solaris: Administration Guide—Introduction*
- *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components*
- *IBM DCE Version 3.1 for Solaris: Administration Commands Reference*

- *IBM DCE Version 3.1 for Solaris: Problem Determination Guide*

## Print and Order Books

### IBM DCE Publications

In addition to the hardcopy editions of the *IBM DCE Version 3.1 for Solaris: Quick Beginnings* and *IBM DCE Version 3.1 for Solaris: Release Notes* IBM supplies PostScript files on the CD-ROM for each of the online DCE 3.1 documents for those customers who want the option of having printed documentation.

### OSF DCE Publications

For printed books that are not specific to a particular product, OSF publishes the following DCE publications through The Open Group:

- Open Software Foundation. *Introduction to OSF DCE*
- Open Software Foundation. *OSF DCE User's Guide and Reference*
- Open Software Foundation. *OSF DCE Administration Guide: Core Components*
- Open Software Foundation. *OSF DCE Administration Guide: Extended Services*
- Open Software Foundation. *OSF DCE Administration Reference*
- Open Software Foundation. *OSF DCE Application Development Guide*
- Open Software Foundation. *OSF DCE Application Development Reference*

Although not written specifically for Solaris products, these publications might provide helpful information. These OSF DCE publications are available through bookstores and mail order companies. The following mail order company is in no way associated with IBM, and IBM makes no claim about the services this company provides:

O'Reilly & Associates, Inc., 103A Morris Street, Sebastopol, CA 95472.
Phone: 800-988-9938 (US and Canada), 707-829-0515,
FAX: 707-829-0104 between 7 am and 6 pm PST weekdays.
Internet: order@ora.com

The following O'Reilly books might also be useful:

- Hu, Wei. *DCE Security Programming* , 1st. ed. Sebastopol, CA: O'Reilly & Associates, 1994.
- Rosenberry, Ward. *Understanding DCE* , 2nd. ed. Sebastopol, CA: O'Reilly & Associates, 1993.
- Shirley, John. *Guide to Writing DCE Applications* , 2nd. ed. Sebastopol, CA: O'Reilly & Associates, 1994.

## Using DCE 3.1 for Solaris Documentation

The DCE 3.1 for Solaris product includes user, administration, and application development documentation that is accessible online. The documentation is provided as HTML files that are viewable with any browser that supports HTML 3.2.

See "Appendix A. Online Documentation" on page 85 for more information about the filesets that must be installed to access the DCE for Solaris online documentation.

### Viewing the HTML Documentation

Users with graphic interfaces can use a Web browser such as the Netscape Navigator browser, which is included with the Solaris operating system, to read the DCE documentation HTML files. The Netscape Navigator browser provides hypertext linking, navigation utilities, a hypertext index, graphical display of artwork, search and print facilities, a bookmark function, and an NLS-enabled online help utility. See the Solaris documentation for information on installing the Netscape Navigator browser.

If you have installed the documentation files locally, use your Web browser to view the DCE HTML documentation by opening the file:

**/opt/dcelocal/docs/html/en_US/index.html**

**Note: en_US** can be substituted with the appropriate locale name.

If you have DCE Web Secure installed and configured into a Netscape Web server, go to the URL:

**http://***servername***/dceweb**

On that Web page click **dce docs**.

If you have installed the documentation files and DCE Web Secure is installed and configured, use your Web browser remotely to view the DCE HTML documentation at the following URL:

**http://<server-name>/dcedoc/en_US**

**Note: en_US** can be substituted with the appropriate locale name.

## Printing the PDF Books

If you prefer hardcopy documentation, a set of PDF files are included on the product CD. You can print these books directly from the CD. Go to the location **/opt/dcelocal/docs/pdf/en_US/** and select the PDF file that you want to send to your printer. See “Appendix A. Online Documentation” on page 85 for a listing of the publications and their file prefixes.

**Note: en_US** can be substituted with the appropriate locale name.

# Part 4. Appendixes

# Appendix A. Online Documentation

The following table identifies the documents by file prefix:

| Prefix | Title |
|---|---|
| apgstyle | *IBM DCE Version 3.1 for Solaris: Application Development Guide—Introduction and Style Guide* |
| dirsrv | *IBM DCE Version 3.1 for Solaris: Application Development Guide—Directory Services* |
| appdev | *IBM DCE Version 3.1 for Solaris: Application Development Guide—Core Components* |
| appref | *IBM DCE Version 3.1 for Solaris: Application Development Reference* |
| pdg | *IBM DCE Version 3.1 for Solaris: Problem Determination Guide* |
| admingd | *IBM DCE Version 3.1 for Solaris: Administration Guide—Core Components* |
| admintro | *IBM DCE Version 3.1 for Solaris: Administration Guide—Introduction* |
| dceintro | *IBM DCE Version 3.1 for Solaris: Introduction to DCE* |
| comref | *IBM DCE Version 3.1 for Solaris: Administration Commands Reference* |
| solquick | *IBM DCE Version 3.1 for Solaris: Quick Beginnings* |

The following files are contained in the Online Documentation package: **IDCEenUSd:**

Directory: /opt/dcelocal/docs/html
Directory: /opt/dcelocal/docs/html/en_US
Directory: /opt/dcelocal/docs/html/en_US/APGSTYLE
Directory: /opt/dcelocal/docs/html/en_US/DIRSRV
Directory: /opt/dcelocal/docs/html/en_US/APPDEV
Directory: /opt/dcelocal/docs/html/en_US/APPREF
Directory: /opt/dcelocal/docs/html/en_US/PDG.
Directory: /opt/dcelocal/docs/html/en_US/ADMINGD
Directory: /opt/dcelocal/docs/html/en_US/ADMINTRO
Directory: /opt/dcelocal/docs/html/en_US/DCEINTRO
Directory: /opt/dcelocal/docs/html/en_US/COMREF
Directory: /opt/dcelocal/docs/html/en_US/SOLQUICK
File: /opt/dcelocal/docs/html/en_US/index.html
File: /opt/dcelocal/docs/html/en_US/masthead.gifl
File: /opt/dcelocal/docs/html/en_US/backgr.jpg
File: /opt/dcelocal/docs/html/en_US/README
File: /opt/dcelocal/docs/html/en_US/APGSTYLE/APGSTYLE.TAR.Z
File: /opt/dcelocal/docs/html/en_US/DIRSRV/DIRSRV.TAR.Z

File: /opt/dcelocal/docs/html/en_US/APPDEV/APPDEV.TAR.Z
File: /opt/dcelocal/docs/html/en_US/APPREF/APPREF.TAR.Z
File: /opt/dcelocal/docs/html/en_US/PDG/PDG.TAR.Z
File: /opt/dcelocal/docs/html/en_US/ADMINGD/ADMINGD.TAR.Z
File: /opt/dcelocal/docs/html/en_US/ADMINTRO/ADMINTRO.TAR.Z
File: /opt/dcelocal/docs/html/en_US/DCEINTRO/DCEINTRO.TAR.Z
File: /opt/dcelocal/docs/html/en_US/COMREF/COMREF.TAR.Z
File: /opt/dcelocal/docs/html/en_US/SOLQUICK/SOLQUICK.TAR.Z
Directory: /opt/dcelocal/docs/pdf
Directory: /opt/dcelocal/docs/pdf/en_US
File: /opt/dcelocal/docs/pdf/en_US/README
File: /opt/dcelocal/docs/pdf/en_US/apgstyle.pdf
File: /opt/dcelocal/docs/pdf/en_US/dirsrv.pdf
File: /opt/dcelocal/docs/pdf/en_US/appdev.pdf
File: /opt/dcelocal/docs/pdf/en_US/appref.pdf
File: /opt/dcelocal/docs/pdf/en_US/pdg.pdf
File: /opt/dcelocal/docs/pdf/en_US/admingd.pdf
File: /opt/dcelocal/docs/pdf/en_US/admintro.pdf
File: /opt/dcelocal/docs/pdf/en_US/dceintro.pdf
File: /opt/dcelocal/docs/pdf/en_US/comref.pdf
File: /opt/dcelocal/docs/pdf/en_US/solquick.pdf

# Appendix B. DCE Web Secure for Solaris Advanced Configuration

These instructions cover advanced configuration for DCE Web Secure. These steps assume that your installation and basic configuration were successful.

Advanced configuration involves manually changing the **magnus.conf** and **obj.conf** files for your Netscape Web server. You should be knowledgeable in the syntax and context of these files before making any changes and ensure that you save a backup copy of the files should your configuration changes not perform to your expectations. Consult your Netscape documentation or the Netscape home page for more information on the Netscape configuration files.

## Authenticated Path Configuration

You can specify additional paths that need DCE credentials before access is granted by using the **auth-path** parameter on the PathCheck directive. The path specified is evaluated against translated paths resulting from NameTrans directives in the **obj.conf** file.

You can specify auth-path for the following reasons:

- The path contains DCE-enabled CGI programs.
- The path needs authentication with a DCE keytab file rather than through basic authentication, if used in conjunction with the **keyfile** parameter.

## Keyfile Configuration

When a DCE administrator wants users to access paths that require DCE credentials, such as to a directory containing DCE-enabled CGIs, and wants the user to operate under a specific DCE userid, the DCE administrator can create a keytab file and specify that it be used for a particular path.

DCE Web Secure allows the use of keytab files by adding the **keyfile**=″keytab-file-path″ and **keyfile-user**=″user-in-keytab-file″ parameters to the dce-restrict PathCheck directive in the Web server's **obj.conf** file.

## Summary of Advanced Configuration Syntax

```
PathCheck fn="dce-restrict" <auth-path="path">
 <keyfile="keytab-file-path"
keyfile-user="user-in-keytab-file">
```

The **dce-restrict** PathCheck directive can specify valid combinations of the following optional arguments:

**auth-path**
: Specifies a path prefix that describes objects in the file system that need DCE credentials before access is granted.

**keyfile**
: Indicates that authentication on objects that match the associated auth-path should be performed through the specified keytab file. The keyfile must be an existing, valid DCE keytab file. The keyfile parameter must always be used in conjunction with the keyfile-user parameter.

**keyfile-user**
: Specifies a valid DCE principal in the keyfile keytab file. The keyfile-user parameter must always be used in conjunction with the keyfile parameter.

## Examples of Advanced Configuration

The following example shows multiple **dce-restrict** PathCheck directives that might reside in a Web server's **obj.conf** file.

```
PathCheck fn="dce-restrict" auth-path="/mycgidir"
PathCheck fn="dce-restrict" auth-path="/keytest/cgi"
      keyfile="/var/keyfile" keyfile-user="cgi_server_dceid"
PathCheck fn="dce-restrict"
```

The first statement is an example of how a CGI application might be configured so that DCE authentication is provided when CGI programs in **/mycgidir** are executed. The second statement is an example of how to specify a keytab file for authentication. For example, the following statements were used to set up the keytab file:

```
$ su (Netscape-server-userid)
$ rgy_edit
rgy_edit> kta -p cgi_server_dceid -pw (random-password) -f /var/keyfile
rgy_edit> exit
```

To test whether the keyfile is set up correctly, these statements were used:

```
$ su (Netscape-server-userid)
$ dce_login cgi_server_dceid -k /var/keyfile
```

The third statement shows the initial configuration of DCE Web Secure.

# Appendix C. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make

improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department LZKS
11400 Burnet Road
Austin, TX 78758
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written.

These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. *1999*. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

IBM
AIX
RISC System/6000

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

Windows NT is a trademark of the Microsoft corporation.

Other company, product, and service names may be trademarks or service marks of others.

# Index

## Special Characters



## A



## B



## C

## D



## F



## G



## I



## K



## L



## M



## N



## O



## P



## Q



## R



## S

## T



## U



## W

Part Number: CT772NA

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

CT772NA