Porting Applications to CDE 6

If you want to port an OPEN LOOK or Motif application to CDE, you need to consider the development environment transition as well as the graphical user interface (GUI) transition. The OpenWindows and CDE development environments are different in many ways, but similar in others. This chapter compares and contrasts the two development environments, and discusses recommended and optional CDE integration.

Integrating into the CDE Development Environment	*37*
Comparing OpenWindows and CDE Development Environments	*39*

Note – For a high-level understanding of the CDE development environment, read the *Common Desktop Environment: Programmer's Overview.*

Integrating into the CDE Development Environment

As noted in "Levels of CDE Desktop Integration" on page 3, CDE defines three levels of desktop integration. Recommended and optional integration require changes to your code to implement any of the functionality within these categories. However, your application will be more integrated with the desktop depending on how much CDE functionality you adopt.

Recommended Integration

CDE contains components and guidelines to use so that your application will integrate well with other applications on the desktop:

- Help system
- ToolTalk® messaging system
- Drag and drop
- Internationalization
- Standard Font Names
- Error Messages
- User Customization Issues

Optional Integration

The following CDE components enable you to leverage services provided by the desktop for achieving specialized tasks:

- CDE Motif control widgets
- Data typing
- Action invocation
- Workspace manager
- Terminal emulator widget
- Text editor widget
- Calendar
- Desktop Kornshell

CDE Code to Learn From

This section discusses the ways in which CDE provides code for you to either use or learn from to develop CDE applications.

Application Builder

Use Application Builder (App Builder) to build a simple CDE application. Then generate code from it to see what the code looks like. This is a good way to learn how to use CDE Motif and some of the CDE functionality.

Demo Code

CDE contains demo source code that can considerably ease your application porting tasks.

Each development environment component has a demo directory in `/usr/dt/man`. The demo directory contains an example program that uses the component's APIs. Read the demo code to learn how to incorporate the component behavior into your application. In some cases you can copy and paste code from the demo into your application.

The `/usr/dt/man/template` directory contains a demo program that integrates most of the CDE components that comprise basic and recommended integration functionality. This template demo illustrates how to write a simple application that is well integrated with the CDE desktop.

Comparing OpenWindows and CDE Development Environments

This section compares and contrasts some of the features of the OpenWindows and CDE environments.

Features Exclusive to CDE

Desktop Kornshell

Provides a way to engage in graphic user interaction through shell scripts. See the *Common Desktop Environment: Desktop Kornshell User's Guide* for detailed information on Desktop Kornshell.

Workspace Manager

The CDE Workspace Manager provides support for multiple workspaces. Each workspace is a *virtual screen*. Most desktop applications can run as expected without knowledge of the Workspace Manager.

Commonalities

This section contrasts some of the aspects that OpenWindows and CDE development environments have in common.

X Server

CDE is a desktop that is based on the same X server as OpenWindows. For example, you can access display PostScript™ (DPS), XIL™, and XGL™ from CDE.

GUI Application Builders

The OpenWindows Developer's Guide (Devguide) is a tool that helps produce the GUI for OPEN LOOK applications. Devguide has code generators that produced XView and OLIT code. The latest version of Devguide produces Motif code, too.

If you have used Devguide to build an application, the CDE Application Builder (App Builder) will feel very familiar to you. The palette is visually similar. Much of the Devguide functionality is retained, such as the Build and Test feature and the ability to build projects.

Application Builder produces CDE Motif code. It also enables you to integrate some of the desktop services into your application; for example, drag and drop, ToolTalk messaging, sessioning, help, and internationalization.

Drag and Drop

The underlying basic functionality of OpenWindows drag and drop and CDE drag and drop are similar. Both versions of drag and drop contain general purpose APIs that provide the same outcome, from a user's perspective.

However, CDE drag and drop also provides a *convenience* API, that serves two purposes:

- It is easier to use than the general purpose APIs
- It defines and implements data transfer policies

OpenWindows has conventions for drag and drop policies, but it is up to the developer to implement the policies. If you use drag and drop in your OPEN LOOK application, much of the data transfer code you wrote can be condensed by using the CDE convenience APIs.

ToolTalk

OpenWindows ToolTalk is upwardly compatible with the CDE ToolTalk Messaging Service. In addition, the ToolTalk Messaging Service provides:

- Two standard ToolTalk protocols known as *message sets*
- `ttsnoop`

  The `ttsnoop` utility is a debugging tool that helps monitor messages that are sent between your application and other applications.
- `tttrace`

  The `tttrace` utility is a debugging tool that helps monitor ToolTalk API calls.

See the *Common Desktop Environment: Programmer's Overview*, the *Common Desktop Environment: ToolTalk Messaging Overview*, and the `ttsnoop(1)` and `tttrace(1)` man pages for more information on the CDE extensions of ToolTalk.

Typing

The OpenWindows *classing engine* identifies the characteristics, or attributes, of files. The classing engine specifies attributes such as print method, icons, and open commands for specific file types. The classing engine consists of two parts:

- A database that stores file type names and attributes
- A collection of routines that query the database

The CDE *data typing* and *actions* form the analog of the classing engine. The data typing mechanism consists of two tables (DATA_ATTRIBUTES and DATA_CRITERIA) that specify attributes such as icons, actions, and commands for specific file types. DATA_CRITERIA corresponds to the classing engine's File name space. DATA_ATTRIBUTES corresponds to the classing engine's Type name space. The actions field in the DATA_ATTRIBUTES table corresponds to the former print and open methods in the classing engine. In CDE, it acts as a reference to another table called ACTION, and is greatly enhanced over the former classing engine methods.

6