TriTeal Enterprise Desktop (TED™) 4.0

Programmer's Overview

TriTeal™

Printed in USA

Part No. TED009004000

Printing History
August 1995 ... First Edition
Printed in the U.S.A

Sales Information and Technical Support

TriTeal Corporation
2011 Palomar Airport Road, Suite 200
Carlsbad, CA 92009
USA

Tel: (619) 930-2077
Fax: (619) 930-2081
Support E-Mail: support@triteal.com
Sales E-Mail: info@triteal.com
World Wide Web: http://www.triteal.com/

Please
Recycle

Contents

Preface

The *TriTeal Enterprise Desktop: Programmer's Overview* provides a high-level view of the TriTeal Enterprise Desktop development environment and the developer documentation set. Read this book first before starting application design and development.

Note – TriTeal Enterprise Desktop is an implementation of CDE. If you wish to build an application for the TriTeal Enterprise Desktop, it should be consistent with the CDE guidelines defined by the *TriTeal Enterprise Desktop: Style Guide and Certification Checklist*.

Note – In this manual, the terms *TriTeal Enterprise Desktop* and *the desktop* are used interchangeably.

Outside of the Preface, this manual omits the *TriTeal Enterprise Desktop* prefix when referring to a TriTeal Enterprise Desktop development or run-time environment manual. For example, the *TriTeal Enterprise Desktop: Programmer's Overview* is referred to as the *Programmer's Overview.*

Who Should Use This Book

Read the *Programmer's Overview* if you are:

- An application developer who wants to develop a new TriTeal Enterprise Desktop application, or integrate an existing OSF/Motif®, OPEN LOOK® or Indigo Magic® application into the desktop.
- A manager, architect, or project lead interested in designing a project involving applications that will run on the TriTeal Enterprise Desktop.

For the remainder of this manual, OSF/Motif is referred to as Motif®.

How This Book Is Organized

The Programmer's Overview is divided into two parts. Part I contains an architectural overview of the TriTeal Enterprise Desktop, including high-level information on both the run-time and development environments. Part II contains information useful to know before developing an application, and describes the development environment components.

This section provides brief descriptions of the chapters and appendixes contained in this manual.

Chapter 1, "Architectural Overview," presents an overview of the TriTeal Enterprise Desktop architecture.

Chapter 2, "Development Environment Considerations," discusses information you should know about the environment before you start to develop an application.

Chapter 3, "Developing an Application," presents information specific to developing a TriTeal Enterprise Desktop application, such as naming conventions and guidelines to follow.

Chapter 4, "Portability and Maintenance," discusses issues pertaining to writing portable and maintainable applications.

Chapter 5, "Basic Application Integration," summarizes how to make your application launch-integrated (that is, started by double-clicking an icon on the desktop).

Chapter 6, "Recommended Integration," provides overviews of all components and guidelines that you should use so your application has the same look and feel as, and interoperates well with, other TriTeal Enterprise Desktop applications.

Chapter 7, "Optional Integration," provides overviews of the components to incorporate into your application as needed for added functionality.

Appendix , "CDE Motif," describes the differences between Motif 1.2.3 and CDE Motif.

Appendix B, "Component and Guideline Reference," lists in alphabetical order all development environment components and guidelines, with associated library, header files, and documentation.

Related Books

For information on Motif, see:

- *OSF/Motif Programmer's Guide*, Release 1.2, by Open Software Foundation, 11 Cambridge Center, Cambridge, MA 02142, published by PTR Prentice Hall, Englewood Cliffs, NJ 07632.
- *OSF/Motif Programmer's Reference*, Release 1.2, by Open Software Foundation, 11 Cambridge Center, Cambridge, MA 02142, published by PTR Prentice Hall, Englewood Cliffs, NJ 07632.
- *OSF/Motif Reference Guide*, by Douglas A. Young, published by PTR Prentice Hall, Englewood Cliffs, NJ 07632.
- *OSF/Motif 1.2 Style Guide*, by Open Software Foundation, 11 Cambridge Center, Cambridge, MA 02142, published by PTR Prentice Hall, Englewood Cliffs, NJ 07632.

Note – The *TriTeal Enterprise Desktop: Style Guide and Certification Checklist* is an extension of the *OSF/Motif 1.2 Style Guide* to the TriTeal Enterprise Desktop.

- *OSF Application Environment Specification (AES) User Environment Volume*, Revision C, by Open Software Foundation, 11 Cambridge Center, Cambridge, MA 02142, published by PTR Prentice Hall, Englewood Cliffs, NJ 07632.

Motif 1.2 IEEE Std 1295 standard, which you can order from:

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855

For information on POSIX, see the IEEE Std 1003.1-1990 standard, which you can order from:

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855

For information on Xlib, see:

- *Xlib Programming Manual* for Version 11 (Volume 1) by Adrian Nye, published by O'Reilly & Associates, Sebastopol, CA, 95472.
- *Xlib Reference Manual* for Version 11 (Volume 2), published by O'Reilly & Associates, Sebastopol, CA, 95472.

For information on Xt, see:

- *X Toolkit Intrinsics Programming Manual*, (Volume 4) by Adrian Nye and Tim O'Reilly, published by O'Reilly and Associates, Sebastopol, CA 95472.
- *X Toolkit Intrinsics Reference Manual,* (Volume 5) edited by Tim O'Reilly, published by O'Reilly and Associates, Sebastopol, CA 95472.

For more information on ToolTalk®, see:

- *The ToolTalk Service: An Inter-Operability Solution*, published by SunSoft Press and PTR Prentice Hall, Englewood Cliffs, NJ 07632, ISBN 0-13-088717-X.
- *ToolTalk and Open Protocols: Inter-Application Communication*, by Astrid Julienne and Brian Holtz, published by SunSoft Press and PTR Prentice Hall, Englewood Cliffs, NJ 07632, ISBN 013-031055-7.

In addition to the *Programmer's Overview*, the development environment documentation set consists of:

- *TriTeal Enterprise Desktop: Style Guide and Certification Checklist*
- *TriTeal Enterprise Desktop: Application Builder User's Guide*
- *TriTeal Enterprise Desktop: Programmer's Guide*
- *TriTeal Enterprise Desktop: Help System Author's and Programmer's Guide*
- *TriTeal Enterprise Desktop: ToolTalk Messaging Overview*
- *TriTeal Enterprise Desktop: ToolTalk Programmer's Guide*
- *TriTeal Enterprise Desktop: Internationalization Programmer's Guide*
- *TriTeal Enterprise Desktop: Desktop KornShell (dtksh) User's Guide*
- Online man pages

For more information on these development environment books, see the following section, “Development Environment Documentation.”

The run-time environment documentation set consists of:

- *TriTeal Enterprise Desktop: User’s Guide*
- *TriTeal Enterprise Desktop: Advanced User’s and System Administrator’s Guide*
- *TriTeal Enterprise Desktop: TEDvision Guide*
- *TriTeal Enterprise Desktop: TEDfax Guide*
- *TriTeal Enterprise Desktop: TED Enhancements*
- Online help volumes

Note – The *Advanced User’s and System Administrator’s Guide* contains information to help you integrate an application into the desktop.

Development Environment Documentation

This section provides an overview of each manual—except for the *Programmer’s Overview*—in the developer documentation set.

TriTeal Enterprise Desktop: Style Guide and Certification Checklist

The *TriTeal Enterprise Desktop: Style Guide and Certification Checklist* provides application design style guidelines and the list of requirements for CDE application-level certification. These requirements consist of the Motif Version 1.2 requirements with CDE-specific additions.

The checklist describes keys using a model keyboard mechanism. It assumes that your application is being designed for a left-to-right language environment in an English-language locale. Wherever keyboard input is specified, the keys are indicated by the engravings on the Motif model keyboard. Mouse buttons are described using a virtual button mechanism to better describe behavior independent from the number of buttons on the mouse.

This book provides information to assist the application designer in developing consistent applications and behaviors within the applications.

TriTeal Enterprise Desktop: Application Builder User's Guide

The TriTeal Enterprise Desktop Application Builder (also called App Builder) is an interactive tool for developing TriTeal Enterprise Desktop applications. It provides features that facilitate both the construction of an application graphical user interface (GUI) and the incorporation of the desktop's many useful desktop services (such as Help, ToolTalk, Drag and Drop). The *TriTeal Enterprise Desktop: Application Builder User's Guide* explains how to create an interface by dragging and dropping "objects" from a palette. It also explains how to make connections between objects in the interface, how to use the application framework editor to easily integrate desktop services, how to generate C code, and how to add application code to the Application Builder output to produce a finished application.

TriTeal Enterprise Desktop: Programmer's Guide

The *TriTeal Enterprise Desktop: Programmer's Guide* has two parts. Each part provides a detailed description of elements of the TriTeal Enterprise Desktop, a conceptual diagram, and a task-oriented description of how to use each element, complete with code examples.

Part I, "Recommended Integration," provides an overview of basic integration, and describes how to integrate new applications with the Session Manager, fonts and drag and drop. It also discusses displaying error messages.

Part II, "Optional Integration," describes how to integrate new applications with the Workspace Manager, TriTeal Enterprise Desktop Motif widgets, actions, data types and Calendar.

The *Programmer's Guide* provides an introduction to the application program interfaces (APIs) for the components referred to in the descriptions of Parts I and II above, with cross-references to the relevant man pages. Details are covered in the man pages.

TriTeal Enterprise Desktop: Help System Author's and Programmer's Guide

The *TriTeal Enterprise Desktop: Help System Author's and Programmer's Guide* describes how to develop online help for application software. It covers how to create help topics and how to integrate online help into a Motif application.

The audience for this book includes:

- Authors who design, create, and view online help information.
- Developers who want to create software applications that provide a fully integrated help facility.

This book has four parts. Part I describes the collaborative role that authors and developers undertake to design application help. Part II provides information for authors organizing and writing online help. Part III describes the Help System application programmer's toolkit. Part IV contains information for both authors and programmers about preparing online help for different language environments.

TriTeal Enterprise Desktop: ToolTalk Messaging Overview

The *TriTeal Enterprise Desktop: ToolTalk Messaging Overview* describes the ToolTalk components, commands and error messages offered as convenience routines to enable your application to conform to Media Exchange and Desktop Services message set conventions. This manual is for developers who create or maintain applications that use the ToolTalk service to interoperate with other applications.

The *ToolTalk Messaging Overview* does *not* describe general ToolTalk functionality. For detailed information about the ToolTalk service, refer to *The ToolTalk Service: An Inter-Operability Solution*. For tips and techniques to help make using ToolTalk easier, read *ToolTalk and Open Protocols: Inter-Application Communication*. Both of these books are listed in "Related Books" on page xi.

TriTeal Enterprise Desktop: ToolTalk Programmer's Guide

The *TriTeal Enterprise Desktop: ToolTalk Programmer's Guide* describes the application programming interface (API) components, commands and error messages of the ToolTalk service and how you modify your application to send and receive ToolTalk messages.

The ToolTalk service supports several messaging styles. A sender can address a ToolTalk message to a particular process, to any interested process, to an object, or to an object type. Message senders are not concerned with the locations of processes and objects in any network; the ToolTalk service finds receiving processes and objects.

TriTeal Enterprise Desktop: Internationalization Programmer's Guide

The *TriTeal Enterprise Desktop: Internationalization Programmer's Guide* provides information for internationalizing an application so that it can be easily localized to support various languages and cultural conventions in a consistent user interface.

Specifically, this guide:

- Provides guidelines and hints for developers on how to write applications for worldwide distribution.
- Provides an overall view of internationalization topics that span different layers within the desktop.
- Provides pointers to reference and more detailed documentation. In some cases, standard documentation is referenced.

This guide is not intended to duplicate the existing reference or conceptual documentation, but rather to provide guidelines and conventions on specific internationalization topics. It focuses on internationalization topics and not on any specific component or layer in an open software environment.

TriTeal Enterprise Desktop: Desktop KornShell (dtksh) User's Guide

The *TriTeal Enterprise Desktop: Desktop KornShell (dtksh) User's Guide* describes how to create Motif applications with Desktop KornShell (`dtksh`) scripts. It contains several example scripts of increasing complexity, in addition to the basic information a developer needs to get started.

This guide is intended for developers who find a shell-style scripting environment suitable for a particular task. It assumes a knowledge of KornShell programming, Motif, the Xt Intrinsics, and, to a lesser extent, Xlib.

What Typographic Changes and Symbols Mean

The following table describes the type changes and symbols used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. system% You have mail.
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* be root to do this.

Part 1 — TriTeal Enterprise Desktop Architectural Overview

Architectural Overview 1

Note – TriTeal Enterprise Desktop is an implementation of CDE. If you wish to build an application for the TriTeal Enterprise Desktop, it should be consistent with the CDE guidelines defined by the *TriTeal Enterprise Desktop: Style Guide and Certification Checklist*.

This chapter presents a high-level architectural view of the TriTeal Enterprise Desktop. For details regarding the desktop run-time environment, consult the run-time documentation set and the online help volumes. For details regarding the desktop development environment components, see Chapter 6, “Recommended Integration,” Chapter 7, “Optional Integration,” Appendix , “CDE Motif,” the development environment documentation set and the online man pages.

Conceptual Overview

The TriTeal Enterprise Desktop architecture has many cross-process relationships. The three-process relationship of an X client, a window manager and the X Window System™ server seems simple by comparison. The area covered by the TriTeal Enterprise Desktop is broad, but the layering in the system is not as rigorous as that of Motif, Xt and Xlib. The relationships between high-level system components are diverse and extensible. This chapter groups the technologies to illustrate that each desktop component fits into an overall whole. The TriTeal Enterprise Desktop can be divided into:

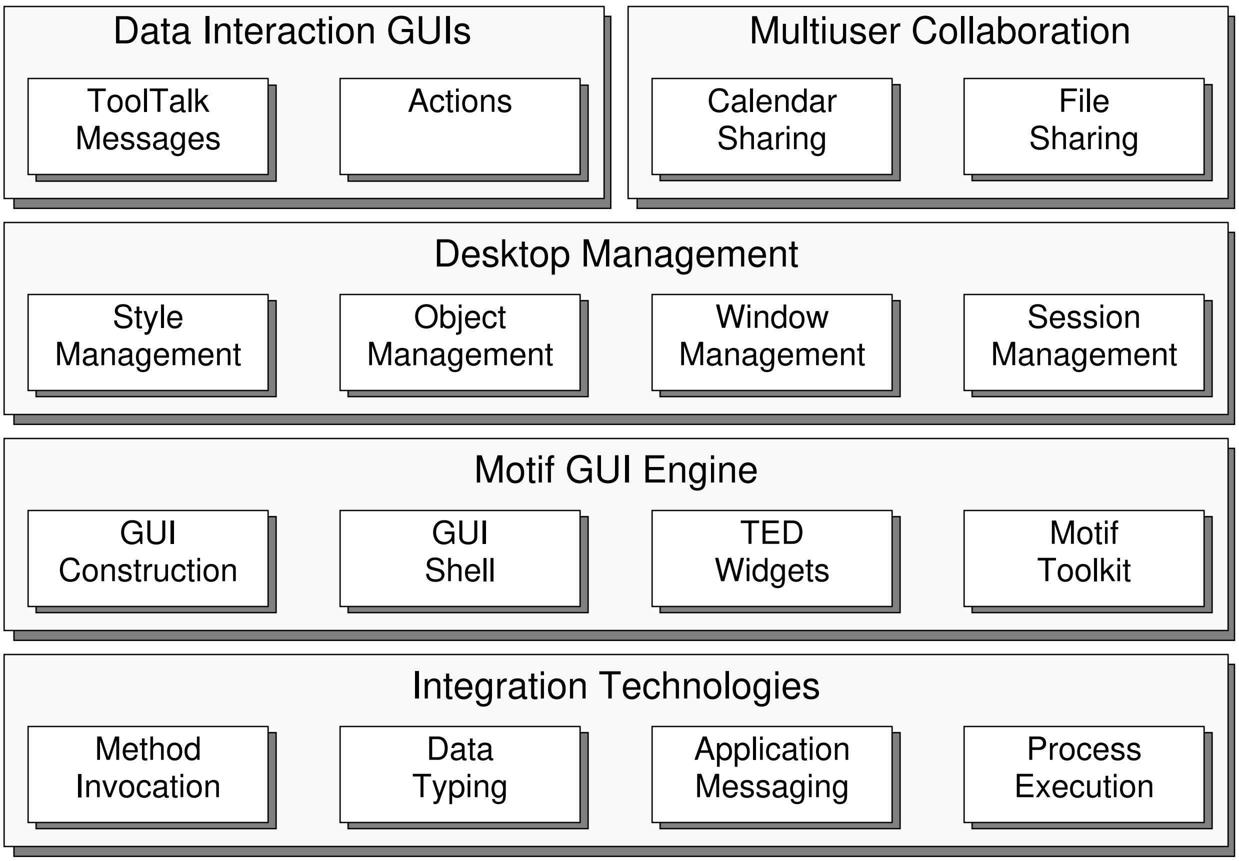

Figure 1-1 Conceptual overview of TriTeal Enterprise Desktop

- Data interaction graphical user interfaces (GUIs)—Application-level components that are available for user interaction, invocable by other applications. Think of these as programming components at a larger granularity than widgets.
- Multiuser collaboration—Defines and uses application program interfaces (APIs) that enable collaboration between users on the network, particularly in the areas of calendar management, network resource naming, and network file sharing.

- Desktop management—Provides components that negotiate the visual relationships between entities on the desktop. These include the following: Window Manager, Workspace Manager, Session Manager, Application Manager, File Manager, Style Manager, and the Front Panel.
- Motif GUI engine—Includes those components that implement the controls available to the user and includes the CDE Motif toolkit, additional widgets, a GUI shell (Desktop KornShell), and a GUI construction tool (Application Builder).
- Integration technologies—Represent technologies that do not generate GUIs, but are used as infrastructure by the rest of the desktop. These technologies include process execution control, application messaging (mechanism and protocols), data typing, and method invocation.

Data Interaction GUIs

The TriTeal Enterprise Desktop supplies a registration service, the ToolTalk Messaging Service, that enables an application to find an available service provider. ToolTalk provides the low-level messaging infrastructure. A companion mechanism, called the actions system, provides a consistent abstraction layer on top of both the traditional UNIX™ command-line interface to applications and the TriTeal Enterprise Desktop-recommended ToolTalk interface to applications. Actions, as semantic entities, are exposed to the end user through higher levels of software. Both actions and ToolTalk are discussed in more detail in "Integration Technologies" on page 16.

The desktop contains components that are available through action and ToolTalk APIs. Examples include GUIs to show a view of a directory, submit a print job, view the contents of the Trash Can, edit some text, show help information, compose a calendar appointment, and compose a mail message.

You can also incorporate actions and ToolTalk message support into your application so that the application-specific services they supply are available to the desktop and other applications. Particularly, applications should provide the composition, viewing, editing, and printing services for both proprietary and standard format data. This way, applications that are coded to accept an extensible set of data types automatically gain more capabilities as more *media* handlers are added to the system. The TriTeal Enterprise Desktop File Manager, Front Panel, and Mailer attachment GUI are examples of such applications.

Media is used as a generic term for anything that can be presented to the user to convey information. The desktop provides media handlers for appointments, mail messages, mail folders, text, icons, and help data. Vendors have extended the desktop with additional media handlers, including PostScript™, many kinds of image file formats, and audio data.

Multiuser Collaboration

While the ToolTalk and action mechanisms encourage cooperation between applications, the desktop also defines cross-user collaboration technologies. This means distributed access to shared user data. The desktop has defined some basic sharing mechanisms and has also built on top of existing mechanisms.

An example of building on an existing mechanism is the remote procedure call (RPC) client/service implementation of calendar management. The desktop provides a client-side library and API, RPC protocol, and daemon/service that enables users to share appointment information. (The API is being standardized through X.400 Application Programming Interface Association (XAPIA) to enable a cross-UNIX, PC, and palmtop calendar standard.) The RPC protocol enables a user to browse and directly edit another user's calendar. Access is controlled by a user-specific access control mechanism. Calendars are tied to hosts, and a calendar's data is maintained by a host-specific daemon. The desktop names calendars through a `user@host` format.

The TriTeal Enterprise Desktop uses conventional distributed file systems to name files that are sharable on the network. To provide an interface that is independent of the distributed file system, the desktop provides an API to translate host-relative file names into locally expressible file names. Although the desktop is based on the NFS® system, it can be ported to run on top of other distributed file systems. Using the desktop file-name mapping API, an opaque file name object can be constructed and passed between desktop clients across the network and resolved in a host-specific way. Also, to simplify the programming task and end user metaphor, TriTeal Enterprise Desktop applications should present remote file references as local file paths.

One of the fundamentals of building multiuser collaboration applications is the ability to share files. The conventions for naming network files, in conjunction with a ToolTalk file-sharing mechanism called *file scoping*, enable multiuser collaboration through file sharing. File scoping is more than a mechanism for simple, exclusive access control. Cooperating clients can use file-scope access to negotiate for access to files. For example, an application that has exclusive access to a file could ask whether the user was done with the file when another application wanted to gain exclusive access to the file.

Desktop Management

The physical metaphor associated with the TriTeal Enterprise Desktop is loosely one of a user sitting in a chair surrounded by a bank of desks (workspaces). As the user swivels the chair (by clicking a push button on the Front Panel), another desk becomes accessible. On each desk, the following is available:

- A collection of drawers (File Manager views) in which folders (directories) and reports (files) are organized.
- A collection of papers in use on the desktop (windows). Some papers are pushed out of the way (as icons), but are within easy reach.
- Continuous display (through Front Panel icons) of a clock, the date, an indication of new mail, and an indication of something in the trash can.
- Direct access (through Front Panel buttons) to an appointment book (Calendar), a pad of paper (Text Editor), a terminal (emulator), a mail box (Mailer), a printer (Print Manager), office lighting controls (Style Manager), a list of electronic agents (Application Manager which includes applications like TEDfax and TEDvision, and Front Panel personal tool box), and a guide book (Help).

The user drags and drops objects to change their location and make copies of them. By dropping objects on services, the user gains assistance with appointment scheduling, editing, mail composition, faxing, posting in the network, printing, and so on.

Session Management

The state of the desktop can be remembered. At a later time, and perhaps at a different X display station, the state of the desktop can be re-created. A session is a snapshot of the state of a user's desktop at a point in time. The TriTeal Enterprise Desktop supports two sessions from which the user can choose:

- Home session—A snapshot of the desktop state that reassembles in the same way each time it is started.

- Current session—The state of a desktop saved at logout time.

The TriTeal Enterprise Desktop Session Manager coordinates these activities, but applications are responsible for saving their own state.

The desktop uses the X11R5 Interclient Communication Conventions style of session management. This consists mostly of conventions for setting properties on top-level windows. The desktop extends this by providing a facility that allocates specific files into which applications can store their state. A command-line flag then points to this file when the application is restarted. Applications that maintain multiple top-level windows must save the state of each of them.

A session is associated with a particular user. In the TriTeal Enterprise Desktop, the Login Manager is responsible for initial user login. The Login Manager is an alternative GUI for the UNIX login program. Normally, it checks the entered password with the user's registered password. However, vendors can provide authentication schemes tuned to their platform.

The Login Manager is network-aware. When faced with an X display that is normally served by host A, the user can log into the user's desktop by running a session from host B that has full access to the user's normal set of files and services on host B. This is possible by Login Manager acting as the desktop's X11 Display Manager (XDM). The XDM Control Protocol (XDMCP) is used between X11 window servers and XDMs on the network. The Login Manager displays its login window or host chooser window on any X11 server requesting either XDM service. This makes the TriTeal Enterprise Desktop a good match for use with XDMCP-aware X terminals.

For connections to the X server, the desktop uses the X magic cookie scheme to control access. If a user on some host machine can read a certain file within a session owner's home directory, then access to the X server is granted. An alternative to this per-user authorization is per-host authorization. This is useful for installations supporting pre-X11R4 clients, which will be unable to connect to X servers using the X magic cookie scheme.

X resource files are handled in the context of TriTeal Enterprise Desktop sessions as follows: a set of TriTeal Enterprise Desktop default resources is merged with a host version of this file, followed by the user's `$HOME/.Xdefaults` file, followed by a session-specific file of resources that have changed through user interaction with the Style Manager. The result is stored in the RESOURCE_MANAGER property of the root window. To enable fine-grain customization, the C preprocessor is run on resource files.

Application Management

One of the obstacles preventing end users from taking full advantage of the network environment is the difficulty of accessing remote applications. The TriTeal Enterprise Desktop provides conventions for:

- Installation of applications so that they can be run remotely.
- User navigation of available applications.
- Execution of remote applications.

The user can browse the collection of available applications with a GUI tool called Application Manager. Applications can be dragged onto the desktop for easier access. Even remote applications are started by a simple double-click, hiding the network location of a running application. The user is not aware of any distinction between local and remote applications.

This network transparency is accomplished by installing applications on network hosts designated as *application servers*. The parts of the installation relevant to the desktop require placing certain files in conventional places in the application's installation hierarchy. The application server maintains a list of applications that it is serving. Each host on the network maintains a list of the application servers on the network that it queries when a user logs into the desktop. This process is referred to as *application gathering*. It results in a dynamically-generated file hierarchy of actions arranged in folders. (Actions represent operations that end users can invoke, including starting applications.)

The TriTeal Enterprise Desktop Application Manager provides a specialized view of the file system for the end user. Applications are arranged into groups and groups can be nested (such as in a directory hierarchy). Your application’s installation script associates the application to a group. This association can be overridden by the system administrator as part of application server configuration. The set and arrangement of the actions shown through the Application Manager is a system resource that is typically shared between multiple users. Users cannot modify this view.

The user can drag an icon from the Application Manager onto the desktop, File Manager, Front Panel, and so on. The associated action remains valid as long as the gathered application that it refers to remains valid. Because actions represent a form of abstraction and indirection, the actual location of the application can change over time. This change remains transparent to the end user (this is explained further in “Method Invocation” on page 18). The user double-clicks on an action icon to invoke it.

Object Management

The TriTeal Enterprise Desktop captures some object-oriented system attributes without being dependent upon a completely object-oriented infrastructure. The desktop provides graphic onscreen images that the user can pick up and move about, dropping them anywhere it makes semantic sense. These are viewed as *objects* by the user. The File Manager promotes the object abstraction by providing a graphical way to browse and modify file and directory objects within the file system. It also provides a GUI to invoke actions. When the user selects a file, the actions that are defined for the selected type of file are presented to the user.

Objects managed by desktop-based applications do not have to be file-based; in-memory buffers can represent desktop objects, too. The TriTeal Enterprise Desktop Mailer handles Multipurpose Internet Mail Extensions (MIME) messages by displaying attachments to a message as icons in a scrollable panel. These are objects that behave just like file-based objects during activities such as drag and drop. The user can drag between the File Manager and the Mailer. Applications that use drag and drop should maintain this important user model by supporting both file-based and buffer-based objects. The desktop drag-and-drop API and protocol make this straightforward.

Window Management

The Window Manager is essentially the Motif 1.2 window manager with extensions to provide the Front Panel GUI and workspace abstraction.

The Front Panel can be thought of as a graphic version of the root window menu supported by many window managers. It can also be thought of as a tuned object manager in which common objects are readily available to the user. The Front Panel can show dynamic system information, and it enables the user to invoke actions and system functions. The user dynamically customizes the Front Panel by dragging and dropping action icons from the Application

Manager and File Manager onto subpanels. Applications can come equipped with special configuration files that extend the Front Panel, possibly defining drop behavior, drop zone animation feedback, and so on. The user can optionally install these configuration files depending on customization preferences. Figure 1-2 displays a typical desktop Front Panel.

Figure 1-2 Typical Front Panel.

Workspaces are abstractions supported by the Window Manager that can be thought of as virtual desktops. Application windows exist within one, some, or all available workspaces. The user usually determines which workspaces an application window exists in as part of the user's customization. You should rarely use the workspace API other than to explicitly designate in which workspace your application appears on session restart. In general, do not place your application within multiple workspaces, because this overrides the user's prerogative.

Style Management

The Style Manager enables users to customize their desktop using a GUI. Users are shielded from advanced concepts, such as X resources, for most common customization options. Style Manager provides controls for desktop-wide properties that adjust backdrops, keyboard settings, mouse settings, screen saver options, window management, workspace management, and session management. These properties either do not affect applications directly or indirectly affect them through the X server or window manager.

You, as an application developer, are more directly influenced by font choices, color choices, and input device mappings. The Motif toolkit and the TriTeal Enterprise Desktop handle many of these settings transparently for widgets. However, your application will appear more integrated with the rest of the desktop if it responds to user font and color preferences. Applications that directly interact with the mouse will feel more integrated with the rest of the desktop if they are consistent with other applications; for example, by using the same mouse button double-click minimum interval value (`multiClickTime` resource).

To accommodate differences between platform vendor's display technology and available font sets, the TriTeal Enterprise Desktop defines font aliases that are indirect names to actual font names. Use these aliases in the same way as the rest of desktop uses them.

The Style Manager provides the user with color selection options to adjust the desktop color scheme. This color information is private to the TriTeal Enterprise Desktop. Applications doing widget subclassing can indirectly access some of the color scheme by looking at inherited background pixel values. A call to `XmGetColors()` generates 3-D shadow colors.

The TriTeal Enterprise Desktop does not dictate color usage for static colors, such as those used within icons. For these situations, however, your application should attempt to use the colors offered by the TriTeal Enterprise Desktop Icon Editor, to enhance color sharing.

Motif GUI Engine

Think of the Motif toolkit as the GUI engine of the desktop. This section discusses CDE Motif, CDE widgets, and alternative modes of Motif programming.

Note – TriTeal Enterprise Desktop is an implementation of CDE and is based on the CDE Motif and CDE Widgets. If you wish to build an application for the TriTeal Enterprise Desktop, it should be consistent with the CDE guidelines defined by the *TriTeal Enterprise Desktop: Style Guide and Certification Checklist*.

TriTeal Enterprise Desktop Motif Toolkit

The CDE Motif toolkit is Motif 1.2.3 with bug fixes, enhancements, and some new features. You must explicitly set resources to enable the new features. Functional additions include file selection box GUI modifications, different default settings of existing resources (primarily to lighten up the default border widths), color management enhancements, internationalization of error messages, and minor usability fixes (some of which have the effect of easing migration of OPEN LOOK users to the TriTeal Enterprise Desktop).

CDE Motif and Motif 2.0 are also highly compatible. Most functions put into CDE Motif have been introduced into Motif 2.0. As a result, developers have compiled their applications with CDE Motif, relinked to Motif 2.0, and ran the applications successfully. Widget subclassing that has not followed Motif 1.2 subclassing guidelines designed to shield programs from widget size changes are likely to fail.

A drag-and-drop convenience layer has been added on top of the Motif 1.2 drag-and-drop API. In addition, the TriTeal Enterprise Desktop uses the Motif 1.2 preregister drag feedback protocol. A drop site drag manager process keeps track of visible drop zones on the desktop. This data is used by a drag source client process to manage drag feedback interaction. Limited drag time validation of drop zones is followed by full validation at drop time, with *snap-back-to-source* animation if the drop fails.

CDE Motif includes a GUI style guide and certification checklist that has substantially expanded on the Motif 1.2 style guide. Additions affect the input models, window management, and GUI design principles.

CDE Motif Widgets

CDE Motif provides two types of widgets that are not available in Motif 1.2.3:

- Low-level control widgets:
  - SpinBox—A text field and arrow button widget.
  - ComboBox—A text field and list box widget.
  - MenuButton—A menu that doesn't need to be in a row column widget.

  These were added primarily to help you port applications from a Microsoft® Windows or OPEN LOOK environment. The SpinBox and ComboBox widgets have equivalents in Motif 2.0.

- Rich and full-featured widgets:
  - Terminal Emulator widget—Useful for applications designed to mix the best of a command-line user interface with a GUI.
  - Editor widget—Available for embedding a more full-featured plain text editor than that available from the Motif Text widget.
  - Help widgets—Handle navigation and interaction with application help volumes.

Help is delivered with an application in the form of Semantic Description Language (SDL) files that have been compiled from *HelpTag*, a form of Standard Generalized Markup Language (SGML) files. The Help system features mixed text and graphics, hyper links, dynamic reformatting of text, and structured navigation capabilities.

GUI Shell

The TriTeal Enterprise Desktop includes Desktop KornShell (`dtksh`), an interpreted scripting language alternative to C programming of the Motif toolkit. Desktop KornShell includes selected frequently-used CDE APIs, along with Xt, and Xlib APIs. You must use a compiled language to access the full power of the environment. However, you can write Desktop KornShell scripts that participate in desktop integration activities such as drag and drop, session management, and ToolTalk messaging.

If you are comfortable with shell programming, you may prefer to use Desktop KornShell (`dtksh`) for modest programming tasks because it is:

- Well suited to system-administration-type applications because the shell commands intermix easily with GUI control.
- Good for putting a GUI control program on top of character-based applications because the shell environment handles character-based interaction in a natural way.
- A good way to deliver instruction-set-independent programs to a heterogeneous collection of hosts. For example, use the TriTeal Enterprise Desktop Mailer to attach a script to a message that the recipient simply double-clicks to invoke.

GUI Construction

The easiest way to produce a TriTeal Enterprise Desktop application, and perhaps the fastest, is to do almost no Motif toolkit programming at all. Use the TriTeal Enterprise Desktop Application Builder, also known as App Builder, to construct the GUI control portion of your application. App Builder focuses on making default widget behavior easy to access. It does this by hiding many of the more esoteric resources that are available on most widgets. App Builder also makes it as easy to incorporate desktop integration infrastructure into your application, including drag and drop, session management, and ToolTalk messaging.

App Builder maintains the user interface state in Builder Interface Language (BIL) files. A code generator takes the BIL files and produces Motif toolkit code. App Builder can also generate User Interface Language (UIL) files.

As you make changes to your application's user interface, App Builder merges your custom code with the code it generates. Generated code is a good source of example code, even if you do not using App Builder to maintain your application's GUI state.

In addition, nonprogrammers can use App Builder to produce an application GUI prototype. The prototype can roll forward to programmers for the production phase of development.

Integration Technologies

TriTeal Enterprise Desktop technologies discussed thus far have been directly involved with putting a GUI onto the screen. The integration technologies described in this section are underlying infrastructure, not GUI providers.

Process Execution

To provide a network-leveraging environment, the TriTeal Enterprise Desktop provides the Sub Process Control (SPC) mechanism to start, manage, and collect results from applications running on a remote host. A remote host installs an SPC daemon that serves as the remote end of a socket-based control mechanism. This control mechanism tries to maintain the illusion that the remote process is a local *child* to the *parent* process. Authentication of the user that owns the parent process is based upon the ability of the parent process to write a `setuid` file to the user's home directory *and* the ability of the child process to read the result.

The SPC API and associated control programs are private to the TriTeal Enterprise Desktop. Actions represent the public API for running applications remotely.

Application Messaging

The ToolTalk Messaging Service is the application messaging mechanism for the TriTeal Enterprise Desktop. Application messaging addresses inter-application control and cooperation for applications working on behalf of a single user. The ToolTalk session daemon is a local message-routing process whose control scope typically corresponds to that of the X server. This means that clients within a session issue requests, the ToolTalk session manager finds or starts some client within a session that is able to handle the request, and the ToolTalk session daemon tracks the request until completion.

The desktop provides two standard ToolTalk protocols known as *messages sets*. A message set contains a number of messages that can be exchanged between a sender and a handler process. These messages are grouped together because they describe related requests and notices. The sender and recipient may be within the same process or on different hosts. Message sets have associated utility functions that allow you to concentrate on the semantics of the protocol without getting involved in the low-level messaging details. Some of the message set functions enable you to defer to default behavior with almost no work on your part.

Desktop Message Set

The Desktop Message Set encompasses three areas. The first is windowing behavior. The second involves file access and short term file life cycle control. The third is specific to applications that have extension languages and is not generic enough to warrant library support.

Media Message Set

The Media Message Set allows an application to be a container for arbitrary media, or to be a media player/editor that can be driven from such a container. The Media message interface allows a container application (such as Mailer or File Manager) to compose, display, edit, or print a file or buffer of an arbitrary media type, without understanding anything about the format of that media type. ToolTalk routes a container's requests to the user's preferred tool for the given media type and operation. This includes routing the request to an already-running instance of the tool if that instance can best handle the request.

Data Typing

The TriTeal Enterprise Desktop, provides the same uniform user interface that CDE provides, to the objects contained on the desktop. To do this, the desktop has a mechanism, called data typing, to determine an object’s type using a set of criteria. The criteria includes properties potentially shared by file-based and buffer-based objects such as name pattern and content pattern. Other criteria are exclusive to files, and include path-name pattern and file permissions. Associated with every desktop type is an extensible set of attributes, including icon name, name template pattern, list of actions suitable for presentation to a user, equivalent type names for other type spaces (for example, MIME type), and a textual description of this type. The *actions and data-types database* stores data criteria and data attributes.

The TriTeal Enterprise Desktop, which is based on CDE, defines (and platform vendors supply), a set of desktop type definitions. Your application should augment the database with both proprietary and public data types at application installation time.

Information is extracted from the actions and data-types through a CDE library API. The data typing API matches an object's properties with the database type criteria to determine the object's desktop type. The matching algorithm uses a set of precedence rules to resolve conflicts.

The CDE type space is defined by the X/Open CDE standard and exists primarily to support desktop-oriented activities such as icon display and action association. The MIME type space is defined by the Internet Engineering Task Force and exists to deal with exchange of mail message parts. A ToolTalk media type space exists in order to match data with handlers, and is a subset of X selection target types defined by the X Consortium. Thus, to do a complete job of type definition, you have to define a CDE type, X selection target, and MIME type. For private CDE types, append the type name to an organization's name. This partitions the name space without need for centralized allocation of types. The TriTeal Enterprise Desktop claims the *Dt* prefix, for *Desktop*.

Method Invocation

A CDE type can be thought of as the class of a desktop object. Using this analogy, actions can be thought of as the methods available on instances of a class. Thus, the actions attribute in a type attribute list describes operations that are available for the type. A single action in the actions and data-types database has multiple parts, many of which are optional. These parts include:

- A description of how to invoke the operation: for example, through ToolTalk, through an execution string passed to the SPC mechanism, from within a terminal emulator, and so on.

- A description of the type of arguments associated with the action. The type of the desktop objects (files and buffers) that it accepts is defined by the actions and data-types database. Actions are polymorphic with respect to data types. For example, the Open action invokes a text editor for arguments that are text files and a graphics editor for arguments that are graphics files.
- A description of the number of arguments, if any, associated with the action.
- An optional indication as to where to carry out the operation: the local machine, a particular remote machine, the machine on which the executable resides, and so on. In addition, these execution locations can be included in a list so that if a host is not available then the next host on the list is tried. This provides a measure of redundancy that can be used to increase the likelihood of application launch, even in the face of remote host unavailability. Thus, actions provide network distribution guidance, implemented either through built-in ToolTalk facilities or through the SPC mechanism directly.
- An optional label, help string, and icon that the user sees when interacting with the action's GUI. These are hints to an application about how to represent the action to the user. These hints may be ignored, as the Front Panel does by ignoring the icon if the Front Panel configuration file supplies an alternative icon.

The collection of actions available to the user is assembled at the same time as the system is collecting type database information. In fact, related action and type information usually reside together in the same file. Desktop-defined, system-administrator-defined (host-specific), and user-defined files are assembled in order into a single (actions and data-types) database, with later definitions taking precedence. This ordering of search path precedence and traversal is used elsewhere by the desktop for such things as help volume and icon file searches.

The actions and data-types database and the File Manager use action files to instantiate actions as file system objects that can be viewed, invoked, moved, copied, and so on. The database contains references to an action’s implementation (for example “run `/usr/bin/app` on machine `net_app_svr`”). However, a representation is needed of an action as an object that the user can directly manipulate. This is achieved by using an object's name, which identifies it as an action to any object manager that is looking for actions. Thus, if there is an executable file named `Dtstyle` and an action named Dtstyle, the File Manager will interpret that file, regardless of its content, as the Dtstyle action reference. In addition, the File Manager uses the action's label as the name that the user sees for this file. Action labels are localizable, whereas action names are programmatic entities that should not be localized.

The good feature about using files simply as pointers into the actions and data-types database is that the underlying implementation can evolve without the user having to do anything. However, one user's actions and data-types database may not match another user's actions and data-types database. Thus, a user cannot exchange an action reference, for example as a mail message attachment, and expect another person to have a comparable definition for that action. Exchanging a Desktop KornShell script is the best solution to this problem.

Actions are useful because they integrate both legacy command-line applications and ToolTalk applications into the desktop as polymorphic, distributed operations on desktop objects.

Part 2 — Development Environment Overview

Development Environment Considerations 2

Note – TriTeal Enterprise Desktop is an implementation of CDE. The TriTeal Enterprise Desktop APIs are based on the CDE APIs. If you wish to build an application for the TriTeal Enterprise Desktop, it should be consistent with the CDE guidelines defined by the *TriTeal Enterprise Desktop: Style Guide and Certification Checklist*.

This chapter discusses general information you should know before starting to use the TriTeal Enterprise Desktop application program interfaces (APIs).

Before you integrate your application into the desktop, you should have a basic understanding of how the desktop works. Install the TriTeal Enterprise Desktop on your platform and familiarize yourself with its features. For an introduction to the desktop, see the *User's Guide* or the *Desktop Introduction* online help volume.

TriTeal Enterprise Desktop Characteristics

The TriTeal Enterprise Desktop provides a productive and comfortable desktop environment for UNIX users. As you develop your application, keep in mind the experience that the TriTeal Enterprise Desktop delivers to its users. Develop your application with the following characteristics in mind, to help make it a powerful, consistent, and predictable part of the TriTeal Enterprise Desktop:

- Hide the complexities of UNIX.

  Because the TriTeal Enterprise Desktop targets end users as its primary customers, providing an application that hides UNIX as much as possible is a key ingredient to a successful product.

- Provide a common look and feel.

  Successful applications in the TriTeal Enterprise Desktop share look-and-feel characteristics with other applications on the desktop. Follow the style and other guidelines (such as the TriTeal Enterprise Desktop standard font names) so that your application encompasses the CDE Motif look and feel.

- Make applications easy to use.

  Provide an easily readable default font size, and provide keyboard accelerators for mouse-oriented actions. Use the desktop online help component to integrate a complete Help system into your application. Basic computer interaction styles should be consistent across platforms wherever possible.

- Take advantage of desktop integration services.

  The TriTeal Enterprise Desktop provides a set of desktop integration services that enable applications to be well-integrated into the desktop. Users benefit because they do not have to know whether an application is running on a local machine or somewhere on the network, or which toolkit (if any) was used to write the application they are running. Provide mechanisms in your application that enable it to be launched from the desktop and to communicate with other TriTeal Enterprise Desktop applications. Use the online Help system to provide users with quick information. Use drag and drop to provide users with a more predictable way to use their systems.

- Design for individual and cultural differences.

  By following the TriTeal Enterprise Desktop conventions and policies, your application will naturally provide for smooth, consistent, and appropriate customization of:
  - Fonts.
  - Color.
  - Keyboard and mouse bindings.
  - Locale-specific configuration files.

  For more about locale-specific configuration files, see the localization section of the *Advanced User's and System Administration Guide*.

Underlying Foundations

To compile an application that uses the desktop APIs, you need:

- TriTeal Enterprise Desktop header files and libraries.
- X11R5 or later header files and libraries.
- ANSI C compiler; or C++ compiler, version 2.0 or later, if you are developing a C++ application.

The resulting binary file must be run in an environment in which matching libraries are installed. The run-time libraries are in `/usr/dt/lib`, and they include the directories listed in Table 2-2 on page 28.

To run properly, TriTeal Enterprise Desktop-based applications require an environment in which some files from the following run-time directories are installed:

- `/etc/dt/*`
- `/var/dt/*`
- The following subdirectories of `/usr/dt`:
  - `app-defaults`
  - `appconfig`
  - `bin`

- `dthelp`
- `lib`

Your application should depend on only those run-time files *explicitly* mentioned in the X/Open XCDE standard. It should *not* depend on files not mentioned in the standard; for example, `/usr/dt/appconfig/icons`. If you build such dependencies into your application, it should be robust enough to run in an environment in which such files are missing or have changed.

For a listing of the minimum run-time environment required to run a TriTeal Enterprise Desktop application, see the `dtfilsys(5)` man page.

The run-time environment includes CDE Motif, which is Motif 1.2.3 with bug fixes and enhancements. (For a description of these enhancements, see Appendix , "CDE Motif.") For more information on the TriTeal Enterprise Desktop run-time environment, see the run-time documentation set.

Running Existing Applications

Existing X Window System-based applications that are not compiled with any TriTeal Enterprise Desktop libraries run under the desktop window manager (`dtwm`) similarly to the way they run under the Motif window manager: they still work. The level of interoperability with the TriTeal Enterprise Desktop follows these guidelines.

Table 2-1 Interoperability of Existing X-Based Applications with the TriTeal Enterprise Desktop

Characteristics of Existing X-based Applications	**Interoperability With Desktop**
Motif 1.2 (and later) drag and drop (using preregister protocol)	Yes
Motif 1.2 (and later) drag and drop (using dynamic protocol)	No[1]
Cut and paste (all applications)	Yes
OPEN LOOK drag and drop	Yes (except multiple-item drag and drop)[2]
ToolTalk Media Exchange and Desktop protocols	Yes

1. This is a Motif 1.2 drag-and-drop dynamic protocol interoperability problem, and it is *not* due to the TriTeal Enterprise Desktop.
2. This is implemented through a protocol translation mechanism in the drop site database manager (`dsdm`).

If you want to recompile and relink an existing Motif application with the CDE Motif shared libraries, the application must be compatible with Motif 1.2.

Libraries and Header Files

Compile TriTeal Enterprise Desktop applications against X11R5 header files and libraries, which reside in vendor-specific locations. Table 2-2 on page 28 lists the locations of all development environment libraries and header files, as subdirectories of `/usr/dt`.

Demo Programs

The `/usr/dt/examples` subdirectories contain source code for development environment component demos, as well as a *template* application. See Table 2-2 on page 28 for a listing of all demo subdirectories.

Each demo subdirectory contains source files for one or more demo programs, along with makefiles for the programs. It also contains a `README` file that describes the demos.

The demo whose source is in the `template` subdirectory is a simple drawing program. This demo illustrates the basic structure of a TriTeal Enterprise Desktop application that is integrated with the desktop. It is internationalized and contains all localized components in a separate subdirectory. You can use the drawing program source as a template for your application.

Man Pages

The TriTeal Enterprise Desktop man pages reside in `/usr/dt/man`. To view them using either the `man()` command or the desktop man page viewer, you must add `/usr/dt/man` to the `MANPATH` environment variable. For example, in your `~/.dtprofile` file, set:

```
MANPATH = $MANPATH:/usr/dt/man
```

For a listing of the `/usr/dt/man` subdirectories and contents, see Table 2-2. For more information on the `man()` command, see the `man(1)` man page.

Development Environment Directory Structure

Table 2-2 lists the top-level directories in the development environment directory structure and lists their subdirectories. (All of the top-level directory names are prefixed by `/usr/dt`.)

Table 2-2 Development Environment Directories in /usr/dt

Directory	Subdirectory	Contents
`examples`		Subdirectories that contain source code for development environment component demo programs.
		README file.
	`dtaction`	Action invocation API demos.
	`dtbuilder`	Application Builder examples.
	`dtcalendar`	Calendar API demos.
	`dtdnd`	Drag-and-drop API demos.
	`dtdts`	Data-typing API demos.
	`dthelp`	Help API demos.
	`dtksh`	`dtksh` example scripts.
	`dtsession`	Session Manager API demos.
	`dtterm`	Terminal Emulator widget API demos.
	`dtwidget`	CDE Motif widgets demos.
	`dtwsm`	Workspace Manager API demos.
	`template`	Template TriTeal Enterprise Desktop application.
	`motif`	Motif 1.2 API demos.
	`motif/clipboard`	`XmClipboard` API demo.
	`motif/dogs`	Widget binary compatibility mechanism demo.
	`motif/draganddrop`	Motif 1.2 drag-and-drop API demo.
	`motif/periodic`	Motif widgets demo.
	`tt`	ToolTalk Messaging Service demos.

Table 2-2 Development Environment Directories in /usr/dt *(Continued)*

Directory	Subdirectory	Contents
`include`		Development environment library header files.
	`csa`	Calendar header files.
	`Dt`	Header files for `DtSvc`, `DtWidget`, `DtHelp`, and `DtTerm`.
	`Mrm`	Motif 1.2 resource manager header files.
	`Tt`	ToolTalk Messaging Service header files.
	`Xm`	Motif 1.2 toolkit header files.
	`uil`	Motif 1.2-callable UIL compiler header files.
`lib`		Library files for `libcsa` (Calendar), `libDtHelp` (Help system), `libDtSvc` (actions, data typing, drag and drop, Session Manager, Workspace Manager), `libDtTerm` (Terminal Emulator), `libDtWidget` (CDE Motif control and editor widgets), `libMrm` (Motif 1.2 resource manager), `libUil` (Motif 1.2-callable UIL compiler), `libXm` (Motif 1.2 toolkit), `libtt` (ToolTalk)
`man`		Development environment man pages.
	`man1`	Client and utility man pages.
	`man3`	API man pages.
	`man4`	Data formats.
	`man5`	Header file and action man pages.

Developing an Application 3

This chapter presents information specific to developing a TriTeal Enterprise Desktop application, such as naming conventions and other guidelines. It introduces levels of integration, which are the guidelines for determining the desktop functionality to incorporate into your application to make it increasingly integrated with the desktop. It also provides an overview of the Application Builder, a tool to simplify TriTeal Enterprise Desktop application development.

Levels of Desktop Integration

Users can run any X11-based application from a shell command line in the TriTeal Enterprise Desktop. If you want to integrate your application into the desktop, however, there are guidelines for you to follow. The TriTeal Enterprise Desktop defines three levels of integration to give you maximum flexibility in designing your application or porting an existing application:

- *Basic Integration*—Enables your application to be launched from the desktop. You do *not* need to change your application code to perform basic integration. See Chapter 5, "Basic Application Integration," for more information.
- *Recommended Integration*—Enables your application to enhance its level of consistency with the desktop. See Chapter 6, "Recommended Integration," for more information.
- *Optional Integration*—Enables you to leverage services provided by the desktop for achieving specialized tasks. See Chapter 7, "Optional Integration," for more information.

For more information on all three levels of integration, see the *Programmer's Guide*. Basic Integration is also discussed in the *Advanced User's and System Administrator's Guide*.

Desktop Naming Conventions

The TriTeal Enterprise Desktop uses naming conventions similar to those used by X and Motif. Desktop clients, desktop libraries, and other desktop components share a common prefix for externally visible names: `dt`, `Dt`, or `DT`. Private desktop structures, functions, and defines (found in the TriTeal Enterprise Desktop code; not for developer use) have an `_dt`, `_Dt`, or `_DT` prefix. Table 3-1 lists the desktop naming conventions.

Table 3-1 Desktop Naming Conventions

Name	Prefix	Example
Desktop clients and utilities	`dt`	`dthelpview`
Resource names and classes	`Dt`	`DtNhelpType, DtCHelpType`
Library names	`Dt`	`libDtHelp`
Include references	`Dt`	`#include <Dt/Help.h>`

Table 3-1 Desktop Naming Conventions *(Continued)*

Name	Prefix	Example
Public function names	`Dt`	`DtCreateHelpDialog`
Public data structure names	`Dt`	`DtHelpDialogCallbackStruct`
Constant names	`Dt`	`DtHELP_NEW_WINDOW`
Environment variables	`DT`	`DTHELPSEARCHPATH`
Private desktop symbols (structures, functions, defines)	`_dt`, `_Dt`, `_DT`	`_DtHelpFunction`, `_DtHELP_DEFINE`

Table 3-2 lists the exceptions to the preceding naming conventions.

Table 3-2 Exceptions to Desktop Naming Conventions

Name	Prefix	Example
CDE Motif	`Xm`	`XmCreateLabel`
`dtksh` Convenience Functions	`Dtksh`	`DtkshAddButtons`

ToolTalk Messaging Service	`tt (for functions)`
	`Tt (for typedefs)`
	`TT (for constants)`
		`tt_open`
		`Tt_message`
		`TT_NOTICE`

X11R5	`X`, `Xt`	`XOpenDisplay`, `XtCreateWidget`

Caution – Do *not* use the prefixes `dt`, `Dt`, `DT`, `_dt`, `_Dt`, `_DT`, `Xm`, `tt`, `Tt`, `TT`, `X`, or `Xt` to define new symbols in your application code. If you do, you might define one that has already been defined—or might be defined in the future—in the TriTeal Enterprise Desktop, ToolTalk, X11R5, or Motif code.

Public and Private Interfaces

If a TriTeal Enterprise Desktop interface is documented in the man pages or the TriTeal Enterprise Desktop documentation set, you can assume that the interface is public unless otherwise stated. An interface is not necessarily public just because it has a header file associated with it. Interfaces that are not documented are private to the TriTeal Enterprise Desktop and are subject to change without notice.

Guidelines for TriTeal Enterprise Desktop Databases

You can find the syntax for the desktop databases, such as those used for actions and data types, in man pages located in the `/usr/dt/man/man4` directory.

For more information on databases, see the *Programmer's Guide*.

Application Initialization and libDtSvc

If your application uses any of the `libDtSvc` APIs (for actions, data typing, drag and drop, Session Manager, or Workspace Manager), it must first initialize the `libDtSvc` library by calling either `DtInitialize()` or `DtAppInitialize()`. Refer to the `DtInitialize(3)` or `DtAppInitialize(3)` man page for more information.

Application Builder

Application Builder (App Builder) is a tool that enables you to easily create the graphical user interface (GUI) for TriTeal Enterprise Desktop applications, without having to write code to call the desktop application program interfaces (APIs). It abstracts the Motif toolkit into simple object palettes and object property sheets. You can use App Builder to construct a wide range of applications, from simple GUI-based programs to complex, integrated systems. It supports User Interface Language (UIL) file import and export to enable you to migrate your application among other Motif-based tools and products.

App Builder is ideally suited for use if you:

- Are not an expert Motif programmer.
- Are not familiar with the CDE Motif widgets.
- Are not familiar with the desktop services (for example, drag and drop, ToolTalk messaging, sessioning, help, and internationalization).
- Want to build your application user interface quickly and be able to change it easily.

- Are working collaboratively with other people to build a single application.

In fact, even if you do not fit into any of the preceding categories, you will likely find App Builder to be appropriate and helpful for your application development.

Using App Builder, you can:

- Lay out the user interface for an application, constructing it piece-by-piece from a collection of objects from the CDE Motif toolkit.
- Define connections between objects to provide application GUI behavior, then use the test mode that enables connections to be tested.
- Add some of the desktop services functionality to your application.
- Edit applications that were previously created using App Builder.
- Merge automatically generated code with hand-generated code.
- Generate C-language source code and associated project files (for example, message catalogs) for the application.

You can compile and invoke your application from within App Builder. You can execute the build, run, and debug cycles all from a common environment without having to exit and restart App Builder.

Demo Programs

You can find the App Builder example programs in `/usr/dt/examples/dtbuilder`. Read the `README` file for detailed information on these programs.

Related Documentation

For more information on Application Builder, see the appropriate man pages, the App Builder help volume, and the *Application Builder User's Guide*.

Portability and Maintenance 4

This chapter contains information you can use to write highly portable applications and use to ensure that your application will be compatible with future TriTeal Enterprise Desktop releases.

Portability Issues

This section presents issues that might affect your application's portability between different platforms that support the TriTeal Enterprise Desktop.

Standards

To be Common Desktop Environment-compliant, your application must follow the Motif 1.2, ANSI-C, and X11R5 standards. If you are developing your application in C++, use C++ version 2.0 or later. No further assumptions are made that you adhere to any standards, such as POSIX, when you write a Common Desktop Environment (or TriTeal Enterprise Desktop) application. Applications that use the desktop application program interfaces (APIs) will be portable to other Common Desktop Environment platforms. However, using POSIX can enhance your software's portability.

The POSIX standard, IEEE Std 1003.1-1990, is entitled *IEEE Standard for Information Technology—Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language],* ISBN 1-55937-061-0.

The Motif 1.2 standard, IEEE Std 1295, is entitled *Standard for Information Technology—X Window System Graphical User Interface—Modular Toolkit Environment.*

For information on ordering IEEE Std 1003.1-1990 and IEEE Std 1295, see "Related Books" on page xi.

Makefiles

Certain libraries that the TriTeal Enterprise Desktop depends on, for example X11R5, are likely to be installed in different locations on different platforms. Include platform-specific references to accommodate this or write a separate makefile for each platform.

Also, the `make` program functionality can differ across platforms. If you want to write just one makefile for your application, use the common `make` functionality used by the platforms to which you want to port your program. Avoid platform-specific `make` features.

The TriTeal Enterprise Desktop requires no additional defined constants (`-D` parameters) to integrate with the desktop. If you do follow a standard, such as POSIX, you might need to compile with additional standard-specific flags. Read about the standard to find out if it includes special compiler considerations.

Each subdirectory of `/usr/dt/examples` contains sample makefile source files for different platforms. These makefiles take into account system differences. In particular, see the `/usr/dt/examples/dtdts directory` for generic examples of makefiles.

Compile Options

To enable your application to find the desktop `include` files, add:

```
-I/usr/dt/include
```

to each makefile's compile line.

Link Options

To enable your application to reference desktop libraries, add:

`-L/usr/dt/lib -l<`*libname1*`> -l<`*libname2*`>...`

to each makefile's link line, where *libname1*, *libname2* are the names of libraries your application needs to reference. You can include as many desktop library names in this line as you want. For example:

```
-L/usr/dt/lib -lDtSvc -ltt -lXm
```

enables your application to reference the Desktop Services, ToolTalk Messaging System, and Motif 1.2 libraries.

File-Naming Conventions

You should limit your application file names, and the file names your application generates, to 14 characters. This will make your application portable to platforms that have this limitation. Some platforms can be configured to have this constraint.

Do not place length limitations on file names that an end user generates.

Display Support

Your application should support the following display options and configurations:

- Monochrome (including black-and-white icons)
- Color (including color icons)
- Small screens, such as VGA (640x480—scale application window or font size to fit completely on the screen)

If you use the Icon Editor to create color icons, your application will share colors with other desktop applications more readily. This helps conserve color cells when running on a Pseudo Color display.

CDE Motif Widget Binary Compatibility Guidelines

Any widget subclass that you implement using the standard Xt APIs that relies on the size of data structures of widgets from which you have subclassed might not be compatible with any new version of Motif or the TriTeal Enterprise Desktop. The reason for this is that new fields may be added to the superclasses in the new version of Motif. An example is that new fields have been added to the `XmManager` and `XmPrimitive` classes in Motif 2.0.

The incompatibility occurs because a subclass must contain compiled-in references to its instance fields that are specified relative to the start address of the widget instance. When you install a new Motif library for a widget whose superclass instance structure has been extended, the compiled-in references will point to the wrong memory location.

To avoid this difficulty, Motif provides a mechanism for defining resources and accessing widget fields that allows you to reference all fields in the instance and constraint structures relative to the start of the widget `part` structure, instead of the overall widget structure. (The overall widget structure includes the superclass part structure.) The mechanism resolves these relative references at run time, when the widget class is first initialized. To resolve references, it factors in the size of the widget's superclass instance structure, which it reads from the currently linked Motif library.

Note – If you implement subclassing, you *must* use the Motif reference-resolving mechanism if you want your application to be binary compatible with future releases of the TriTeal Enterprise Desktop.

For details regarding this Motif mechanism, consult the Motif 1.2 `XmResolvePartOffsets(3x)` and `XmResolveAllPartOffsets(3x)` man pages and the *OSF/Motif Programmer's Reference*. You can find example source code in `/usr/dt/examples/motif/dogs`.

Basic Application Integration 5

Basic application integration is a set of highly recommended tasks you should perform. These tasks *do not* require modification of the source code for your application.

Basic integration does not involve extensive use of the desktop application program interface (API). Therefore, it does not provide other interaction with the desktop, such as drag and drop, session management, ToolTalk messaging, and programmatic access to the actions and data types database.

Basic Integration Features

Basic application integration provides these features for end users:

- A graphical way to locate and start your application on the desktop.

  Your application will provide a desktop *registration package*, and your installation script will automatically register your application.

  Registration creates an application group at the top level of Application Manager. The application group contains an icon the user double-clicks to start the application.

- The ability to recognize and manipulate your application's data files.

  Your application will provide data types for its data files.

  Data typing configures data files to use a unique icon in File Manager to help users identify them. The data files also have meaningful desktop behavior. Two examples are:
  - The user can start your application by double-clicking a data file.
  - Dropping a data file on a desktop printer drop zone prints the file using the appropriate print command.
- Easy font and color selection using Style Manager.

  Your application will change interface fonts and background, foreground, and shadow colors dynamically.

  The desktop defines general interface font and color resources that are used if no corresponding application-specific resources exist.

Basic integration provides these advantages to system administrators:

- Easy installation and registration.

  Upon installation, the application is automatically registered. The system administrator has little or no additional work to do.
- Easy ongoing administration.

  All the desktop's configuration files are gathered in one location. Furthermore, the application can easily be unregistered if, for example, the system administrator wants to update it or to move it to a different application server.

Organization of Basic Integration Information

Most of the tasks involved in basic integration are also performed by system administrators who are integrating an existing application into the desktop. Therefore, most basic integration documentation is located in the *Advanced User's and System Administrator's Guide*.

The *Programmer's Guide* contains a chapter on basic integration. Where appropriate, the chapter refers you to the information contained in the *Advanced User's and System Administrator's Guide*. It also contains additional information specific to application programmers.

Basic Integration Tasks

These are the general tasks involved in basic integration:

- Modify any application resources that set fonts and colors. This allows users to change the application's interface fonts and colors using Style Manager.

  See the section on modifying font and color resources in the *Advanced User's and System Administrator's Guide*.

- Provide printing for your application's data files.

  See the *Programmer's Guide* basic integration chapter for details.

  Some types of printing integration require that you modify your application code. They are optional, and are discussed in the *Programmer's Guide* basic integration chapter because they are closely related to basic integration tasks.

- Create the registration package for your application.

  See this documentation:

  - *Programmer's Guide* basic integration chapter.
  - *Advanced User's and System Administrator's Guide.*

- Modify your application's installation script to install the registration package files and perform the registration procedure.

  See this documentation:

  - *Programmer's Guide* basic integration chapter.
  - The section on registering the application using `dtappintegrate` in the *Advanced User's and System Administrator's Guide.*

Recommended Integration 6

The TriTeal Enterprise Desktop contains components and guidelines to use so that your application will integrate well with other applications on the desktop. This chapter provides an overview of each recommended component and guideline that you should use to enhance your application's level of consistency with the desktop.

Note – In addition to incorporating the components and following the guidelines in this section, you should also follow the basic integration steps outlined in Chapter 5, "Basic Application Integration."

For more information on recommended integration, see the *Programmer's Guide*.

Help System

The TriTeal Enterprise Desktop Help system is a complete system for developing and displaying online help for application software. It enables authors to write online help that includes rich graphics and text formatting, hyperlinks, and access to the Help system from within the application. The Help system provides a programmer's toolkit for integrating the help facilities into an application.

Creating and integrating online help into an application can be done as a collaborative project. Developers design and implement how an application responds to a user's request for help. Authors organize and write the actual help information that is displayed.

The Help system includes:

For Authors

- TriTeal Enterprise Desktop HelpTag markup language—a set of *tags* used in text files to mark organization and content of online help.
- TriTeal Enterprise Desktop HelpTag software—a set of software tools for converting HelpTag files into run-time help files.
- TriTeal Enterprise Desktop Helpview application—a viewer program for displaying online help.

Authors create help topics using the Help tag set and follow Structured Graphic Markup Language (SGML) tagging conventions. SGML markup is the primary data format. When compiled, the run-time distribution format is SGML-compliant.

The Help system also supports non-SGML formats such as UNIX man pages, text files, and text strings.

For Programmers

- `DtHelp` programming library—Application program interface (API) for creating and integrating help windows into your application.

- `DtHelp` widgets—`DtHelpDialog` and `DtHelpQuickDialog` widgets to create help dialog boxes and quick help dialog boxes (these are also part of the Help library).

Library and Header Files

The Help library, `libDtHelp`, provides support for creating and managing help dialogs based on Motif. The `libDtHelp` header files are:

- `Dt/Help.h`
- `Dt/HelpDialog.h`
- `Dt/HelpQuickD.h`

Demo Programs

You can find the Help system demos in `/usr/dt/examples/dthelp`. Read the `README` file for detailed information on the demos.

Related Documentation

For more information on the Help system, see the relevant man pages and the *Help System Author's and Programmer's Guide*.

ToolTalk Messaging Service

The TriTeal Enterprise Desktop defines two standard ToolTalk protocols known as *message sets*. A message set contains a number of messages that can be exchanged between a sender and a handler process. These message are grouped together because they describe related requests and notices. The sender and recipient can be within the same process or on different hosts. Message sets have associated utility functions that enable you to concentrate on the semantics of the protocol without getting too involved in low-level details. Some message set functions enable you to easily defer to default behavior.

The *desktop message set* encompasses three areas:

- Windowing behavior.
- File access and short-term file lifecycle control.
- Application extension languages.

See "Handle Desktop" and "Send Desktop" for information on windowing behavior. See "Desktop File" for information on file access and short-term file lifecycle control. Implementing the `Do_Command` request is specific to the application's extension language and is not assisted by the ToolTalk Messaging Service.

The *media message set* enables an application to be a container for arbitrary media or to be a media player and editor that can be driven from such a container. The media message set enables a container application to compose, display, edit, and print a document of an arbitrary media type, without understanding anything about the format of that media type. The ToolTalk Messaging Service routes a container's requests to the user's preferred tool for the given media type and operation. This includes routing the request to an already-running instance of the tool, if that instance can best handle the request. See "Send Media" and "Handle Media" on page 49.

The ToolTalk Messaging Service provides support for these message sets:

- Handle Desktop

  Handling desktop requests is the most basic level of messaging integration. Any application that sends ToolTalk messages, either through calling `tt_message_send()` or `DtActionInvoke()`, should handle the desktop requests. This enables other applications to set or query things such as your application's current directory, iconic state, and `$DISPLAY`. For further information, see the man pages on `ttdt_open()`, `ttdt_session_join()`, `ttdt_session_quit()`, and `ttdt_close()`.

- Send Desktop

  When an application is started by `ttsession` to handle some ToolTalk request, it is a child of `ttsession` rather than of the request sender. The application will usually be started on the same X display session as the sender, but not necessarily on the same X11 screen or in the same current directory context . If the application is implemented as a server process, it may already be displaying on a particular screen or in a particular directory context.

  Using desktop requests, a handling application can inherit from the sender attributes that might otherwise be inherited through command-line invocation. Use the desktop message set in this way to reset the handler's locale, current working directory, and even `$DISPLAY`. This enables a carefully coded receiving application to come up on the same X11 screen as the sender. A request handler can also find out the request sender's current directory and window geometry. Knowing the window geometry enables the request handler's window to avoid obscuring the request sender's window, if possible. For more information, see the `ttdt_sender_imprint_on()` man page.

- Handle Media

  The ToolTalk Messaging Service enables an editor to easily handle the standard media requests for the media types for which the editor is responsible. For further information, see the man pages for `ttmedia_ptype_declare()`, `ttdt_message_accept()`, `ttmedia_load_reply()`, and `ttmedia_Deposit()`.

- Send Media

  The ToolTalk Messaging Service enables a container to easily send media requests and manage the subsequent document updates sent back by the handler. In those cases in which the container doesn't engage in any ongoing ToolTalk dialog with a media handler, use the actions API instead of directly using these ToolTalk APIs. Equivalent actions (Open and Print) represent a higher level of abstraction that supports the equivalent of ToolTalk and non-ToolTalk aware media handlers.For further information, see the man pages for `ttmedia_load()` and `ttdt_subcontract_manage()`. Note that, in most cases, a container application should perform operations on objects using `DtActionInvoke()` instead of `ttmedia_load()`. See the *ToolTalk Messaging Overview* for a description of how ToolTalk applications can be driven using actions.

- Desktop File

  The ToolTalk Messaging Service makes it easy to send and receive the desktop messages about files. These messages enable applications to coordinate access to files. For further information, see the man pages for `ttdt_file_join()`, `ttdt_file_quit()`, `ttdt_file_event()`, `ttdt_Get_Modified()`, `ttdt_Save()`, and `ttdt_Revert()`.

Examples of applications that already use the ToolTalk Messaging Service include the TriTeal Enterprise Desktop Icon Editor, Mailer, Text Editor, TEDfax, TEDvision, and Calendar. Other parts of the TriTeal Enterprise Desktop use the ToolTalk Messaging Service indirectly by defining actions that send messages.

Library and Header Files

The ToolTalk messaging library is called `libtt`. The `libtt` header files are:

- `Tt/tt_c.h`
- `Tt/tttk.h`

Demo Programs

You can find the ToolTalk Messaging Service demos in `/usr/dt/examples/tt`. Read the `README` file for detailed information on the demos.

Related Documentation

For more information on the ToolTalk Messaging Service, see the relevant man pages and the *ToolTalk Messaging Overview*.

Session Manager

Session Manager supports the ICCCM 1.1 `WM_COMMAND` and `WM_SAVE_YOURSELF` protocols, which permit:

- Your application to save state information at logout.
- Session Manager to restart your application at login.

Session Manager also provides an API to assist your application in saving and restoring its state at logout and login.

Session Manager is responsible for restarting applications at login. To do this, your application must tell Session Manager what command and command-line options are required to restart it. Use Xlib's `XSetCommand()` to set the `WM_COMMAND` property on your application's top-level window.

When Session Manager saves a session, such as at logout, your application might need to save some state information so it can be restored to a similar state. Session Manager can optionally notify your application that the session is being saved. Your application must inform Session Manager that it wants such notification. It does this by registering the `WM_SAVE_YOURSELF` protocol with its top-level window `WM_PROTOCOLS` property and setting up a callback procedure to handle the notification. To do this, use the `XmAddWMProtocols()` and `XmAddWMProtocolsCallback()` functions. Your application should not interact with the user in any way when processing the WM_SAVE_YOURSELF callback. (For example, it should not display a Save As dialog box.) It must set the WM_COMMAND property on its top-level window to notify Session Manager that it is done saving its state.

To enable your application to save state information, use the DtSessionSavePath() function to obtain the full path name of a file in which this information can be saved. At session restore time, use the DtSessionRestorePath() function to obtain the full path name of the state file your application uses to restore its state.

The TriTeal Enterprise Desktop Workspace Manager is responsible for restoring an application's main top-level window (containing the WM_COMMAND) property to the proper workspace, geometry, and icon state. If an application has multiple top-level windows, it is the application's responsibility to restore the states of the other top-level windows. Refer to "Workspace Manager" on page 70 for additional information.

Library and Header Files

The Desktop Services library, libDtSvc, provides access to many desktop APIs, including the one for session management. Include the Dt/Dt.h and Dt/Session.h header files to access the Session Manager API.

Note – If your application uses any of the Session Manager APIs, it must first initialize the libDtSvc library by calling either DtInitialize() or DtAppInitialize(). Refer to the DtInitialize(3) or DtAppInitialize(3) man page for more information.

Demo Programs

You can find the Session Manager demos in /usr/dt/examples/dtsession. Read the README file for detailed information on the demos.

Related Documentation

For more information on Session Manager, see the relevant man pages and the *Programmer's Guide*.

Drag and Drop

The TriTeal Enterprise Desktop provides a drag-and-drop API, that is layered on top of the Motif 1.2 drag-and-drop API, to provide convenient, consistent, and interoperable drag and drop across the desktop. The TriTeal Enterprise Desktop drag-and-drop API makes it easier for developers to implement drag and drop. With drag and drop, users can manipulate objects on the screen directly by grabbing them, dragging them around the display, and dropping them on other objects to change the object's location or perform a data transfer.

Motif 1.2 drag and drop provides low-level drag-and-drop mechanisms; TriTeal Enterprise Desktop drag and drop incorporates policies for those mechanisms.

TriTeal Enterprise Desktop drag and drop consists of an API and protocols to simplify the interface to Motif drag and drop. It implements policies such as the buffer transfer protocol and the drag cursors' appearances. Use the TriTeal Enterprise Desktop drag-and-drop API, (which is based on the CDE API), with its built-in policies, to ensure interoperability through consistency. TriTeal Enterprise Desktop drag-and-drop policies are compatible with standard Motif 1.2 drag-and-drop protocols for text and file name transfers.

TriTeal Enterprise Desktop drag and drop uses the X selection mechanism to transfer data. Suitable targets exist and are registered with the X Consortium. Two desktop applications can agree to transfer data through the text, file name, or data transfer protocols.

The existing Motif 1.2 API for drag and drop is flexible and, therefore, is somewhat difficult for nonexpert developers to use. The TriTeal Enterprise Desktop drag-and-drop API provides some convenience functions that result in an API that is simpler and easier to use:

- Manages the configuration and appearance of drag icons.

  TriTeal Enterprise Desktop drag and drop provides graphics for the default source, state, and operation icons that compose the drag icon in Motif 1.2.

- Defines a buffer transfer protocol.

  Motif 1.2 drag and drop defines protocols for file name and text string only.

- Enables animation upon drop.

  The drop zone can define an animation procedure that is called when the drop completes.

- Provides enumeration of targets for TEXT and FILE_NAME transfers.
- Provides dual registration.

  You can register a text widget as a drop zone for data other than text, while preserving the ability to accept text drops.
- Provides prioritized drop formats.

  The order in which you specify protocols for the drop zone indicates the relative priority of the protocols desired.

Library and Header Files

The Desktop Services library, `libDtSvc`, provides access to many desktop APIs, including that for drag and drop. Include the `Dt/Dt.h` and `Dt/Dnd.h` header files to access the drag-and-drop API.

Note – If your application uses any of the drag-and-drop APIs, it must first initialize the `libDtSvc` library by calling either `DtInitialize()` or `DtAppInitialize()`. Refer to the `DtInitialize(3)` or `DtAppInitialize(3)` man page for more information.

Demo Programs

You can find the drag-and-drop demos in `/usr/dt/examples/dtdnd`. Read the `README` file for detailed information on the demos.

Related Documentation

For more information on TriTeal Enterprise Desktop drag and drop, see the relevant man pages and the *Programmer's Guide*.

Internationalization

The TriTeal Enterprise Desktop is internationalized to support single-byte and multibyte locales. Developers can write internationalized applications that can be easily localized to run on any TriTeal Enterprise Desktop platform.

TriTeal Enterprise Desktop applications (both source and binary) can be localized into regional languages and territories, and across multiple vendors and hardware platforms:

- Latin American
- Western European
- Japanese
- Korean
- Chinese (Traditional and Simplified)

The TriTeal Enterprise Desktop takes advantage of internationalization features in these standards:

- IEEE 1003.2-1992 (POSIX.2 Annex B)
- X Window System, Version 11 Release 5 (Locales and Internationalization Text Functions)
- Motif 1.2 (Internationalizing and Localizing Motif clients)

If you intend to internationalize your application, you must ensure that it supports input and output of multibyte characters. Also, make sure that message catalogs are used and code can be fully localized.

Demo Programs

The drawing program demo in `/usr/dt/examples/template` is internationalized. Read the `README` file for detailed information on this demo.

Related Documentation

For more information on TriTeal Enterprise Desktop internationalization, see the development environment component man pages and the *Internationalization Programmer's Guide*.

Standard Font Names

The standard font names defined by the TriTeal Enterprise Desktop are guaranteed to be available on all Common Desktop Environment-compliant systems. These names do not specify actual fonts. Instead, they are aliases that each system vendor maps to the vendor's best available fonts. If you use only these font names in your application, you can be sure of getting the closest matching font on any Common Desktop Environment-compliant system. These comprise a set of X Window System font names you can use for the most common categories of type designs and styles.

The standard font names are mapped to different fonts on different TriTeal Enterprise Desktop platforms, typically using the X font alias mechanism. This eliminates the problem of having to select from a varying set of fonts on different platforms. It also enables you to make use of the default set of fonts on a particular vendor’s TriTeal Enterprise Desktop implementation.

The TriTeal Enterprise Desktop defines two types of standard fonts: application fonts and interface fonts. Use the application fonts for output produced by your application. Motif widgets and the desktop use interface fonts; do *not* change their default fonts.

Application Fonts

At least six point sizes are available on all TriTeal Enterprise Desktop platforms for each font associated with a Standard Font Name: 8, 10, 12, 14, 18, and 24. XLFD font descriptions for TriTeal Enterprise Desktop fonts look like:

```
-dt-application-*
```

when used where such patterns are valid.

Two of the most common design variations in fonts used to display text are the presence or absence of serifs and the choice between proportional or regularly spaced (monospaced) characters. Combining these two design variations yields four generic font designs:

- Serif proportionally spaced
- Sans serif proportionally spaced
- Serif monospaced
- Sans serif monospaced

Common examples of each of these four designs (in corresponding order) are:

- Times Roman
- Helvetica
- Courier
- Lucida Typewriter

Each of these designs for text fonts typically come in four styles (combinations of weight and slant):

- Plain
- Bold
- Italic
- Bold-italic

The four styles of each of the four design variations yield 16 generic font variations. These 16 generic fonts are among the most commonly used in general desktop computing. For example, Times Roman, Helvetica, and Courier, each in the four style variations, along with the Symbol font, constitute the *Adobe® 13*—the minimum set of fonts built into all PostScript printers.

Your application might not require an exact font family or name, but will need to use, for example, a monospaced font, a sans serif font, or a serif font. You do not have to know the exact font names present on a particular TriTeal Enterprise Desktop platform. The TriTeal Enterprise Desktop standard fonts default to the vendor's selection of the best font of a particular design on the vendor's platform.

Specify the XLFD font names for the standard application fonts your application needs as font resource values in the application's `app-defaults` file. If you do not use these font names, you might need to supply a different `app-defaults` file for each application on each TriTeal Enterprise Desktop platform.

Interface Fonts

Interface fonts are the small set of finely optimized fonts that define the look of the desktop on a particular platform. These fonts cleanly and quickly convey small amounts of information, such as that appearing in window titles, buttons, menus, and text fields.

The desktop and the Motif toolkit widgets use interface fonts. Do not use these fonts directly within your application windows.

The standard interface font names are different from the standard application font names. They, like the application font names, are mapped to different fonts on different TriTeal Enterprise Desktop platforms. Interface fonts come in three styles:

- *System*—Read-only text (used for limited amounts of text, for example, on menus, buttons, and labels).

- *User*—Text the end user enters, or text appearing in objects built from `XmText`-type and `DtTerm`-type widgets.
- *User bold*—Like the User font, but in bold.

Each style comes in seven sizes. Using the Style Manager, users can choose the size of interface fonts they want on their desktop.

Demo Programs

The drawing program demo in `/usr/dt/examples/template` does not specify any of its own interface fonts. It serves as an example of how the TriTeal Enterprise Desktop Motif interface fonts appear. However, this demo does not take advantage of application fonts.

Related Documentation

For more information on standard fonts, see the relevant man pages—particularly `DtStdAppFontNames(5)` and `DtStdInterfaceFontNames(5)` for the list of XLFD font names—and the *Programmer's Guide*.

Displaying Error Messages from Your Application

Applications in the TriTeal Enterprise Desktop follow a common model for presenting error messages and warnings. Users running your application expect messages to be displayed in message footers, error dialog boxes, or warning dialog boxes, with further explanations available in online help, when appropriate.

This section outlines conventions for displaying error messages in your application. Because of the way message text is handled, it is important to follow these error presentation guidelines precisely. For example, casual users who start your application from the Front Panel never see messages that you send to standard error or standard out. In the TriTeal Enterprise Desktop, such messages are directed to log files (`$HOME/.dt/*log`) that many users do not routinely examine or know about.

How to Present Error Messages

Follow these rules when deciding where to tell users about warnings, messages, and error conditions:

- *If this message is informational*, display the text in the message footer of the application. (*Example:* "`MyDoc` file copied.")
- *If this message is about an error or serious warning*—a problem where an operation important to the user has failed—display an error dialog box or warning dialog box.

What Information to Present in Error Dialogs

A good error dialog or warning dialog gives a user the following information:

- What happened (from the user's point of view).
- Why it happened, in simple language.
- How to fix the problem.

Linking Message Dialogs to Online Help

In cases where additional background information is required, or where it takes more than four or five lines of a dialog to completely explain an error, add a button that links the user to the appropriate section of online help.

Related Documentation

For details on displaying error messages in your application and linking message dialogs to online help, see the *Programmer's Guide*.

User Customization Issues

This section presents guidelines to follow when designing your application's user interface.

Color Use

When you design your application's user interface, do not specify color settings that override the default color scheme that the TriTeal Enterprise Desktop provides for Motif and desktop widgets. For application-defined colors, use the following colors to promote sharing with other desktop applications:

- Black
- White
- Red
- Green
- Blue
- Yellow
- Cyan
- Magenta
- Gray (eight shades: #de, #bd, #ab, #94, #73, #63, #42, and #21)

In most cases, you should not specify colors, so that your application uses the colors chosen by the end user in the desktop Style Manager.

Font Use

For your Motif widgets, use the fonts supplied by the TriTeal Enterprise Desktop so that your application's windows look like other desktop client windows and so that users can change the size of these fonts using the Style Manager. If you override the supplied fonts by changing the Motif `fontList` resource specifications, then you must provide additional functionality if you want users to be able to customize the fonts in your application.

Use the fonts from the TriTeal Enterprise Desktop standard application font names to specify—in your `app-defaults` file—resources you use within your application (aside from the ones Motif uses for its widgets). This ensures that your application finds the appropriate fonts on all TriTeal Enterprise Desktop platforms, which makes your application more portable across such platforms. For more information, see "Standard Font Names" on page 54.

Note – The Style Manager only controls fonts for applications written using Motif version 1.2 or later. It will *not* supply correct fonts for Motif 1.1 (or earlier) applications. These applications must specify their own fonts in the `app-defaults` file.

Accessibility

This section provides guidelines for making software applications accessible to people with disabilities.

Physical Disabilities

Provide keyboard access to all application features, such as those usually accessible through menus or drag and drop, to enable people with physical disabilities to more easily use your application.

Visual Disabilities

Follow these guidelines to make your application more accessible to people with visual disabilities:

- Do not hardcode application colors.
- Do not hardcode graphic attributes such as line, border, and shadow thickness. These attributes should scale with font size.
- Do not hardcode font sizes and styles.
- Provide descriptive names for all widgets. In particular, include descriptive names *in your application code* for widgets that do not display labels on the screen; for example, palette items or icons. This often enables screen-reading software to provide descriptive information to blind users.

Hearing Disabilities

Follow these guidelines to make your application more accessible to people with hearing disabilities:

- Never assume that an end user will hear an audible notification.
- Where appropriate, allow end users to choose between audible or visual cues.
- Do not overuse or rely exclusively on audible cues.
- Enable end users to configure frequency and volume of audible cues.

Language, Cognitive, and Other Disabilities

The access guidelines outlined for visual, hearing, and physical disabilities typically benefit end users with cognitive, language, and other disabilities. In addition to those guidelines, include tear-off menus and user-configurable menus for important application features whenever possible.

Mouse Double-Click Speed

For the end user to experience consistency across applications, you should not hardcode double-click durations into your application or `app-defaults` files. This way, when the user changes the double-click time in the Style Manager, your application responds along with the other desktop applications.

Demo Programs

The drawing program demo in `/usr/dt/examples/template` uses the TriTeal Enterprise Desktop's default colors and fonts. This enables the user to customize the colors and fonts in this program by using the Style Manager. Read the `README` file for detailed information on this demo.

Related Documentation

For more information on user customization issues, see the *Style Guide and Certification Checklist*.

Optional Integration 7

The TriTeal Enterprise Desktop components discussed in this chapter enable you to leverage services provided by the desktop for achieving specialized tasks.

Note – In addition to incorporating any components described in this section into your application, you should also follow the basic integration steps outlined in Chapter 5, “Basic Application Integration.” The components discussed in Chapter 6, “Recommended Integration,” are critical to making your application highly integrated with the desktop.

For more information on optional integration, see the *Programmer's Guide*.

CDE Motif Control Widgets

The CDE Motif control widgets are designed to ease porting OPEN LOOK and Microsoft Windows applications to the TriTeal Enterprise Desktop by providing equivalent functionality in CDE Motif. The CDE Motif widgets library `libDtWidget` contains widgets and functions that are used to provide common functionality across all TriTeal Enterprise Desktop applications. The widgets provided include:

- Text field and arrow button widget (`DtSpinBox`)

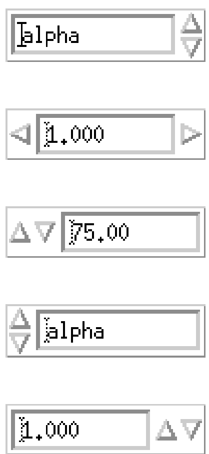

Figure 7-1 Example of text field and arrow button widget (`DtSpinBox`)

- Text field and list box widget (`DtComboBox`)

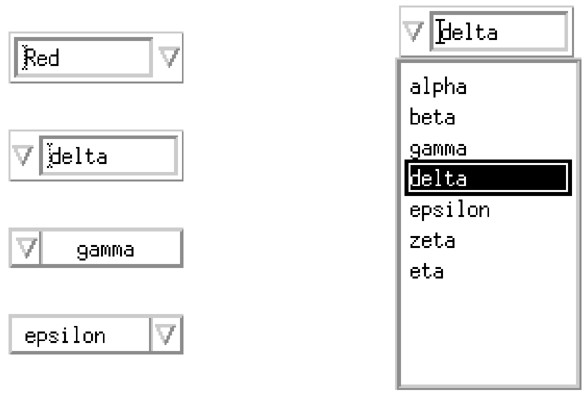

Figure 7-2 Examples of text field and list box widget (`DtComboBox`)

- Menu button widget (`DtMenuButton`)

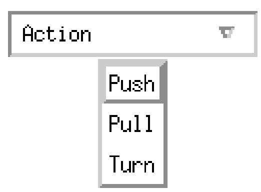

Figure 7-3 Example of menu button widget (`DtMenuButton`)

The CDE Motif Widget library `libDtWidget` supports drivability between TriTeal Enterprise Desktop applications and legacy OPEN LOOK applications. That is, it enables you to write applications that operate the same way OPEN LOOK applications do, in areas such as cut, copy, paste, and mouse-button functionality.

Note – The TriTeal Enterprise Desktop supports all Motif 1.2 widgets. See Appendix , "CDE Motif," for more information on CDE Motif.

Table 7-1 CDE Control Widgets

Widget Name	Description
`DtSpinBox`	`TextField` widget with additional controls for incrementing and decrementing numeric values, or browsing through and selecting from a list of text strings.
`DtComboBox`	Combination of `TextField` and pop-up List widgets that provides a list of valid choices for the `TextField`.
`DtMenuButton`	Command widget that provides menu cascading functionality of an `XmCascadeButton` widget outside of a Menu Bar, or a Menu Pane (a pull-down menu, a pop-up menu, or an option menu).

Compatibility with Motif 2.0

The APIs of the `DtSpinBox` and `DtComboBox` widgets are similar to the Motif 2.0 release of `XmSpinBox` and `XmComboBox` widgets. The APIs are designed so an application can easily switch to the Motif 2.0 version of these widgets. To switch, change the `Dt` names for the class, types, and creation routines to `Xm`. For example, change all occurrences of `DtSpinBox` in your code to `XmSpinBox`. This information is supplied in case you choose to port your application to Motif 2.0, but it is *not* a recommendation that you do so.

Note – The TriTeal Enterprise Desktop does *not* guarantee strict API or binary compatibility between its widgets and the Motif 2.0 widgets.

Library and Header Files

The library `libDtWidget` provides access to the `DtSpinBox`, `DtComboBox`, and `DtMenuButton` widgets. The `libDtWidget` header files for these widgets are:

- `Dt/SpinBox.h`
- `Dt/ComboBox.h`
- `Dt/MenuButton.h`

Demo Programs

You can find the CDE Motif control widgets demos in `/usr/dt/examples/dtwidget`. Read the `README` file for detailed information on the demos.

Related Documentation

For more information on CDE Motif control widgets, see the relevant man pages and the *Programmer's Guide*.

Data Typing

You can define data types when you perform basic integration for your application. This section focuses on a different aspect of data typing: extracting information from the actions and data-types database.

Data typing provides an extension to the attributes of files and data beyond what is provided by traditional UNIX file systems. These extensions include typing and attribute management. Use the data-typing API if your application receives data from external sources and must act on it. For example, your application might want to display an icon or execute an action that has a system-wide definition.

Data typing consist of two parts:

- A database that stores data criteria and data attributes.
- A collection of routines that query the database.

The data-typing system determines a type for a file or byte vector based on a set of criteria. These criteria include its name, permissions, symbolic link value, and contents. The attributes associated with a type describe its user-visible interfaces, including a description, the icon to represent it graphically, and the actions that apply to it. Attributes also exist that name the interchange formats to which the data conforms.

The TriTeal Enterprise Desktop File Manager and TriTeal Enterprise Desktop Mail attachment window use data typing to determine the icon and actions associated with a file. For example, for a C file *filename*`.c`, File Manager uses the `.c` extension to determine the file's data type, `C_SOURCE`. It then uses `C_SOURCE` to determine the icon file name for the icon that it can use for *filename*`.c`.

Database Loading

This section discusses the loading API for the actions and data-types database. The syntax for this and any TriTeal Enterprise Desktop databases a developer defines is discussed in man pages in the `/usr/dt/man/man4` directory.

The external database loading API consists of two functions:

- `DtDbLoad()`
- `DtDbReloadNotify()`

`DtDbLoad()` reads in the actions and data-types database. It determines the set of directories that are searched for database files, and it loads into the database the `.dt` files that are found. Your application must call `DtDbLoad()` before calling any of the routines that query the actions and data-types database.

Use `DtDbReLoadNotify()` to request notification of actions and data-types database reload events. It registers an application's interest in database reload messages.

Your application should reload the database whenever it changes, so that the end user will notice updates without having to restart the application.

Database Queries

To look up an attribute for a data object, you must first determine the type of the object and then ask for the appropriate attribute value for that type. The database query functions enable you to perform operations such as retrieve type data and attributes, free memory, and load and unload the database. These functions are documented in the *Programmer's Guide* and also in man pages in the `man3` directory.

Library and Header Files

The Desktop Services library, `libDtSvc`, provides access to many desktop APIs, including that for data typing. Include the `Dt/Dt.h and Dt/Dts.h` header files to access the data-typing API.

Note – If your application uses any of the data-typing APIs, or loads the actions and data-types database, it must first initialize the `libDtSvc` library by calling either `DtInitialize()` or `DtAppInitialize()`. Refer to the `DtInitialize(3)` or `DtAppInitialize(3)` man page for more information.

Demo Programs

You can find the data typing demos in `/usr/dt/examples/dtdts`. Read the `README` file for detailed information on the demos.

Related Documentation

For more information on data typing, see the relevant man pages and the *Programmer's Guide*.

Action Invocation

The action invocation API enables applications to invoke desktop actions on file or buffer arguments. It chooses a suitable action for the file or buffer based on the `Class`, `Type`, `Mode`, and `Number` of the action arguments. For example, an Open action might invoke an image viewer for GIF files, a word processor for complex documents, and a simple text editor for ordinary ASCII files. Your application need not concern itself with the details of action selection or invocation.

Use the action invocation API so that your application uses the same mechanisms as the rest of the desktop. The user can then experience uniform desktop behavior.

The action functions in the `libDtSvc` library provide a way to invoke desktop actions (such as Open and Print) for files or buffers. They contain parameters that enable you to modify an action's behavior. The action invocation API consists of the following functions:

`DtActionInvoke()` Invokes the specified action on its arguments, which can be files or buffers.

`DtActionLabel()` Provides access to the localizable label string associated with an action.

`DtActionDescription()`
: Returns a string containing the description information associated with the action your application called.

`DtActionExists()`
: a Boolean function that checks whether a given name corresponds to an existing action.

`DtActionIcon()`
: Returns the name of the icon associated with the specified action.

Library and Header Files

The desktop services library, `libDtSvc`, provides access to many desktop APIs, including that for actions. Include the `Dt/Dt.h` and `Dt/Action.h` header files to access the actions API.

Note – If your application uses any of the action invocation APIs, it must first initialize the `libDtSvc` library by calling either `DtInitialize()` or `DtAppInitialize()`. Refer to the `DtInitialize(3)` or `DtAppInitialize(3)` man page for more information.

Demo Programs

You can find the action invocation demos in `/usr/dt/examples/dtaction`. Read the `README` file for detailed information on the demos.

Related Documentation

For more information on actions, see the relevant man pages and the *Programmer's Guide*.

Workspace Manager

The TriTeal Enterprise Desktop Workspace Manager provides support for multiple workspaces. Each workspace is a *virtual screen*. Windows can be placed in a single workspace, all workspaces, or any combination of individual workspaces. Workspaces can be added, deleted, or renamed dynamically by the user.

The Workspace Manager API provides functions for applications that need to know in which workspaces their windows reside, or that need to have some control over how the windows are placed in the workspaces. Additionally, the API enables applications to monitor changes to the overall workspace state, such as which workspace is the current one.

The TriTeal Enterprise Desktop does not require applications to use the Workspace Manager API to run on the desktop. Most desktop applications can run as expected without knowledge of the Workspace Manager. In particular, if your application has a single, main, top-level window and complies with ICCCM 1.1 and Motif 1.2, you do not have to integrate with the TriTeal Enterprise Desktop Workspace Manager for the application to run on the desktop.

However, more complex applications with multiple top-level windows need to use the Workspace Manager API in conjunction with the Session Manager API to properly save and restore the application's state. The Workspace Manager API enables an application to find out which workspaces each of its windows is in. The API also enables an application to display its windows in the correct workspace when the session resumes.

The Workspace Manager API addresses the following tasks:

- Get information on workspaces.
- Get and set the current workspace.
- Notify a client of changes to workspace state.
- Add and remove workspace functions for a client.
- Get and set the workspaces occupied by a client.
- Identify backdrop windows.

All Workspace Manager API functions share the prefix `DtWsm`.

Library and Header Files

The desktop services library, `libDtSvc`, provides access to many desktop APIs, including that for Workspace Manager. Include the `Dt/Dt.h` and `Dt/Wsm.h` header files to access the Workspace Manager API.

Note – If your application uses any of the Workspace Manager APIs, it must first initialize the `libDtSvc` library by calling either `DtInitialize()` or `DtAppInitialize()`. Refer to the `DtInitialize(3)` or `DtAppInitialize(3)` man page for more information.

Demo Programs

You can find the Workspace Manager demos in `/usr/dt/examples/dtwsm`. Read the `README` file for detailed information on the demos.

Related Documentation

For more information on Workspace Manager, see the relevant man pages and the *Programmer's Guide*.

Terminal Emulator Widget

The `DtTerm` widget provides the functionality required to emulate an ANSI X3.64-1979-style terminal emulator (specifically a DEC VT220-like terminal with extensions). The Terminal Emulator widget library, `libDtTerm`, provides the `DtTerm` widget for use in adding a terminal emulator window to a GUI. If you include a terminal emulator in your application, use CDE Motif widgets to add display enhancements to it such as pop-up menus and scroll bars.

The TriTeal Enterprise Desktop Terminal Emulator, which is a part of the run-time environment, is a window that behaves as a terminal, enabling access to traditional terminal-based applications from within the desktop. The `DtTerm` widget is the foundation for the desktop run-time terminal emulator, `dtterm`.

The `libDtTerm` library includes a set of convenience functions to create, access, and support the `DtTerm` widget.

Library and Header Files

The `libDtTerm` library provides a set of widgets based on Motif for designing a terminal emulator or for adding a terminal emulator window to a GUI.

Include the `Dt/Term.h` header file to access `libDtTerm` APIs in your application.

Demo Programs

You can find the Terminal Emulator demos in `/usr/dt/examples/dtterm`. Read the `README` file for detailed information on the demos.

Related Documentation

For more information on the `DtTerm` widget, see the relevant man pages.

For more information on the desktop terminal emulator, see the terminal emulator help volume, the relevant man pages, or the *User's Guide*.

Text Editor Widget

The TriTeal Enterprise Desktop text editing system consists of two components:

- The text editor application, `dtpad`, which provides editing services through graphical, action, and ToolTalk interfaces
- The editor widget, `DtEditor`, which provides a programmatic interface for the following editing services:
    - Cut and paste.
    - Search and replace.
    - Simple formatting.
    - Spell checking (for 8-bit locales).
    - Undo previous edit.
    - Enhanced I/O handling capabilities that support input and output of ASCII text, multibyte text, and buffers of data.
    - Support for reading and writing files directly.

Although the Motif text widget also provides a programmatic interface, applications that want to assure a system-wide uniform editor should use the `DtEditor` widget. The TriTeal Enterprise Desktop Text Editor and Mailer use the editor widget. Use this widget in the following circumstances:

- You need the functionality, such as spell checking, undo, and find/change, that is provided by the `DtEditor` widget.
- You want users to be able to read and write data to and from a file.
- When your program does not need to edit the text while the widget has control of the text.

Library and Header Files

The `DtEditor` widget is in the `libDtWidget` library. The header file is `Dt/Editor.h`.

Demo Programs

A demo containing an example of the `DtEditor` widget (`editor.c`) is in `/usr/dt/examples/dtwidget` directory. Read the `README` file for detailed information on the demo.

Related Documentation

For more information on the Text Editor widget, see the relevant man pages and the *Programmer's Guide*.

Calendar

The TriTeal Enterprise Desktop Calendar comprises the infrastructure and API that enables users to schedule their time and resources in a networked environment. The Calendar GUI is part of the TriTeal Enterprise Desktop run-time environment.

Calendar consists of:

- A daemon that manages the calendar database.
- A calendar and scheduling API that defines a set of high-level functions so that calendar-enabled applications can access the functionality supported by the daemon.
- A library implementation of the calendar and scheduling API.

Additionally, it provides a user interface for both GUI and TTY interaction. The system supports entering, deleting, and modifying calendar entries, as well as browsing and search features. You can access all this functionality through the network.

The development environment provides a library for client access to the Calendar data. It is extensible in that it allows users to define their own calendar entry attributes. The library provides a client callback mechanism for notification of database updates.

The calendar daemon implements the services behind the library of calendar and scheduling API calls. It supports deleting, inserting, and modifying calendar entries. It also manages calendar reminders and supports the creation and removal of the Calendar database. It also provides mechanisms for retrieving Calendar data.

Calendar entry data integrates with the desktop through drag and drop and the ToolTalk messaging interfaces.

The calendar and scheduling API is an implementation of the X.400 Application Programming Interface Association (XAPIA) Calendaring and Scheduling API 1.0. Use the calendar and scheduling API to integrate your application with Calendar, or to develop your own calendar application.

Library and Header Files

The Calendar library, `libcsa`, provides a programmatic way to access and manage Calendar data in a networked environment.

Include the `csa/csa.h` header file to access `libcsa` APIs in your application.

Demo Programs

You can find the Calendar demos in `/usr/dt/examples/dtcalendar`. Read the `README` file for detailed information on the demos.

Related Documentation

For more information on the calendar, see the relevant man pages, the Calendar help volume, and the *Programmer's Guide*.

Desktop KornShell (dtksh)

Desktop KornShell (which is `dtksh`) provides a way to engage in graphic user interaction through shell scripts. The user interface capabilities are based on the CDE Motif widget set, the Xt Intrinsics, and the X11 library.

`dtksh` is a version of `ksh-93` extended to access many X, Xt, Motif, and TriTeal Enterprise Desktop facilities. `ksh-93` is a version of KornShell, the command shell and programming language `ksh`. `dtksh` extends `ksh` to provide support for:

- Access to the CDE Motif widget set from within a shell script.
- Fully localized shell scripts—`dtksh` scripts can use `catopen` and `catgets` commands.
- Access to the TriTeal Enterprise Desktop application Help system.
- Response to session-management Save state directives.
- Access to most of the CDE Services Message Set.
- Access to many of the CDE data-typing API functions.
- Access to most of the CDE action API functions.

Note – TriTeal Enterprise Desktop is an implementation of CDE. TED is based on the CDE Services Message Set, the CDE data-typing API functions, and the CDE action API functions.

Demo Programs

You can find the `dtksh` demos in `/usr/dt/examples/dtksh`. Read the `README` file for detailed information on the demos.

Related Documentation

For more information on `dtksh`, see the relevant man pages and *Desktop KornShell User's Guide*.

CDE Motif — A

Note – TriTeal Enterprise Desktop is an implementation of CDE. If you wish to build an application for the TriTeal Enterprise Desktop, it should be consistent with the CDE guidelines defined by the *TriTeal Enterprise Desktop: Style Guide and Certification Checklist*. Any TED additions to CDE Motif are mentioned in the *TriTeal Enterprise Desktop: TED Enhancements.*

The CDE Motif Toolkit consists of the Motif 1.2.3 widget library with enhancements to existing functionality and bug fixes, as well as some new features. Motif 1.2.3 is a patch of Motif 1.2. The look and feel and APIs for Motif 1.2 and Motif 1.2.3 are the same.

In addition, it provides control widgets for graphical user interface objects not found in Motif 1.2.3. For more information on these widgets, see "CDE Motif Control Widgets" on page 64 and the *Programmer's Guide*.

CDE Motif adds functionality to the Motif 1.2.3 release while maintaining backward binary compatibility. It is source and binary compatible with Motif 1.2 applications. Existing Motif 1.2 applications will compile using CDE Motif. Existing Motif 1.2 binaries will run without modification using CDE Motif.

Features Added to Motif

The TriTeal Enterprise Desktop added the following features to Motif 1.2.3 to support desktop applications:

- Complete localization of toolkit error messages.
- `XmGetPixmap()` and `XmGetPixmapByDepth()` use the environment variable `XMICONSEARCHPATH` or `XMICONBMSEARCHPATH` as the icon search path. If neither of these variables is set, then they use `XBLANGPATH`, which is the Motif 1.2 behavior. See the CDE Motif man page for more information.

Enhancements to Existing Motif Functionality

The CDE Xm library contains minor enhancements to Motif usability to enable better interoperability with OPEN LOOK and Microsoft Windows. The usability enhancements include:

- Optionally allowing mouse button 2 on a three-button mouse to be used to extend the current selection. This is equivalent to the OPEN LOOK Adjust function.
- Allowing Tab to be used to move through a group of `PushButton` widgets and gadgets, `ArrowButton` widget and gadgets, and `DrawnButton` widgets.
- Allowing mouse button 3 to activate a CascadeButton menu.
- Providing three new resources (`pathMode`, `fileFilterStyle`, and `dirTextLabelString`) for the `XmFileSelectionBox` widget, which give it an improved look and feel.
- Enabling interoperability with Microsoft Windows and OPEN LOOK through multiple virtual key bindings.
- Providing visual enhancements to the standard Motif look (see the next section, "Visual Enhancements").

Each of the preceding enhancements can be controlled by a resource: either a widget resource (for `XmFileSelectionBox`) or an application-wide resource (all other cases). The default values for this resource provide behavior and APIs that are identical to that of Motif 1.2. For information on these enhancements and resources, see the `XmDisplay(3x)` and `XmFileSelectionBox(3x)` man pages.

Visual Enhancements

The Common Desktop Environment (or the TriTeal Enterprise Desktop) changed the Motif 1.2.3 look in the following ways:

- RadioBox fill color was changed to show state more clearly.
- RadioBox shape was changed from diamond to circular.
- A check glyph was added to the CheckBox to show state more clearly.
- CascadeButtons and menu items were changed to have an etched-in border when active.
- Thumb was removed from the read-only Scale to distinguish it from the Scale.
- Default shadow thickness was changed to 1 pixel.
- Default highlight thickness was changed to 1 pixel.
- Default PushButton visual that highlights the button inside its default shadow..

For information on these enhancements, see `the XmDisplay(3)`, `XmPushButton(3)`, `XmPushButtonGadget(3)`, `XmToggleButton(3)`, `XmToggleButtonGadget(3)`, and `XmScale(3)` man pages.

Motif Libraries

Use the CDE Motif and X11R5 libraries to develop a CDE Motif-compliant application for the X Window System. The CDE Motif libraries are the Motif 1.2.3 libraries with bug fixes and enhancements.

Motif Library (libXm)

The TriTeal Enterprise Desktop provides all the Motif 1.2.3 header files.

Motif UIL library (libUil)

The Motif User Interface Language (UIL) is a specification language for describing the initial state of a Motif application's user interface. The TriTeal Enterprise Desktop version is essentially unchanged from the Motif version.

Include the `UilDef.h` header file (found in the `uil` directory) to access UIL.

Motif Resource Manager Library (libMrm)

The Motif resource manager (MRM) is responsible for creating widgets based on definitions contained in User Interface Definition (UID) files created by the UIL compiler. MRM interprets the output of the UIL compiler and generates the appropriate argument lists for widget creation functions. Use `libMrm` to access the Motif resource manager. The TriTeal Enterprise Desktop version is essentially unchanged from the Motif version.

Include the `Mrm/MrmPublic.h` header files to access `libMrm` in your application.

Demo Programs

You can find Motif 1.2 demos in `/usr/dt/examples/motif`. These demos do not include any of the CDE Motif enhancements or widgets.

Related Documentation

For more information on Motif, consult the Motif books listed in "Related Books" on page xi. For more information on the enhancements to Motif 1.2.3, see the relevant man pages.

Component and Guideline Reference

B

This appendix alphabetically lists all the TED development environment components and guidelines, along with any associated library and header files, and provides references to associated documentation. In addition to the documentation listed, all components provide man pages, which are located in the `/usr/dt/man` directory. All header files are located in the `Dt` subdirectory of `/usr/dt/include` unless otherwise noted.

Table B-1 Components and Associated Documentation

Component	**Library**	**Header Files**	**Documentation**
Actions and Action Invocation	`libDtSvc`	`Action.h`	*Programmer's Guide*; *User's Guide*; man pages
Application Builder			*Application Builder User's Guide*; *Application Builder Help*
Calendar	`libcsa`	`csa/csa.h`	*Programmer's Guide*; *Calendar Help*
Control Widgets	`libDtWidget`	`ComboBox.h,SpinBox.h, MenuButton.h`	*Programmer's Guide; man pages*
Data Typing	`libDtSvc`	`Dts.h`	*Programmer's Guide*; *User's Guide;*man pages
Drag and Drop	`libDtSvc`	`Dnd.h`	*Programmer's Guide; man pages*

Table B-1 Components and Associated Documentation *(Continued)*

Component	Library	Header Files	Documentation
Desktop KornShell			*Desktop KornShell User's Guide; man pages*
Help System	`libDtHelp`	`Help.h`, `HelpDialog.h`, `HelpQuickD.h`	*Help System Author's and Programmer's Guide;* man pages
Common Desktop Environment Motif	`libMrm`	`Mrm/MrmPublic.h`	*OSF/Motif 1.2 Programmer's Guide; OSF/Motif 1.2 Reference Guide;* man pages
	`libUil`	`uil/UilDef.h`	*OSF/Motif 1.2 Programmer's Guide; OSF/Motif 1.2 Reference Guide;* man pages
	`libXm`	`Xm/*.h`	*OSF/Motif 1.2 Programmer's Guide; OSF/Motif 1.2 Reference Guide;* man pages
Session Manager	`libDtSvc`	`Session.h`	*Programmer's Guide;* man pages
Terminal Emulator Widget	`libDtTerm`	`Term.h`	man pages
Text Editor Widget	`libDtWidget`	`Editor.h`	*Programmer's Guide*; man pages
ToolTalk Messaging Service	`libtt`	`Tt/tt_c.h`, `Tt/tttk.h`	*ToolTalk Messaging Overview;* man pages
Workspace Manager	`libDtSvc`	`Wsm.h`	*Programmer's Guide* man pages

Table B-2 Guidelines and Associated Documentation

Guideline	**Documentation**
Internationalization	*Internationalization Programmer's Guide*
Standard Font Names	*Programmer's Guide*
Displaying Error Messages	*Programmer's Guide*
User Customization (color use, accessibility, mouse double-click speed)	*Style Guide and Certification Checklist*

Index

X