Desktop Integration 1

Desktop integration is the name given to a suite of technologies that allow seamless cooperation and interoperability between applications on the desktop. Desktop integration allows the following:

- Users can select data from one application, and drop it into a different application without regard to format.
- A data object on the desktop can be dragged to the Print Tool where it will be printed in the appropriate format.
- A user can attach an icon representing a desktop publishing file to a mail message. The message receiver could then open the document into the desktop publishing application by simply double-clicking on the icon.
- The development of *groupware*, or applications that allow several people to work simultaneously on a document or program, while the system automatically performs the various housekeeping chores (such as updating files and informing other users of file changes).

Desktop integration lets applications share information and processes with other applications. This sharing results in a higher degree of communication, cooperation, and software productivity.

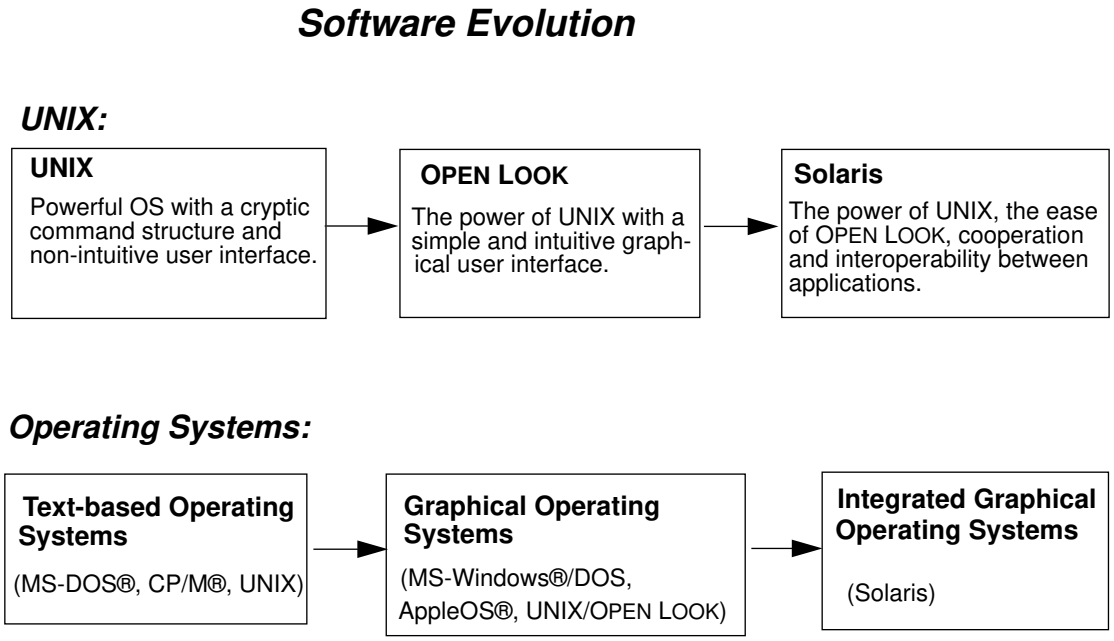
The guide is written for software developers who have previous window programming experience, and who wish to integrate their applications with other applications on the Solaris desktop. This guide presents an overview of

1

desktop integration and its constituent technologies: selections/drag and drop, the Classing Engine, and ToolTalk services. Refer to the technology-specific manuals listed in the Preface for further sources of programming information.

1.1 UNIX Evolution

OpenWindows desktop integration represents an evolutionary step in the growth and maturation of the UNIX® operating system. Although UNIX itself is an extremely powerful operating system, its command line interface is non-intuitive, cryptic, and difficult for most users to master. In recent years the UNIX command line interface has been supplanted by windowing interfaces such as the OPEN LOOK™ graphical user interface, which provides a simpler and more intuitive way to control the system. By the end of 1992, there were more than two thousand OPEN LOOK application programs offered by more than a thousand vendors.



Although OPEN LOOK greatly increases the usability of UNIX, there are some limitations: occasionally pathnames and other elements of command line UNIX need to be manually entered. Also, data exchange is severely limited by

incompatible file formats, and cannot always be performed on files, folders, or other large units of data. Solaris addresses these limitations by combining UNIX, the OPEN LOOK GUI, and the application interoperability tools described in this manual.

1.1.1 SunSoft Terms

OPEN LOOK is the graphical user interface specification adopted by UNIX International and used by SunSoft. OpenWindows, Solaris's window system, implements the OPEN LOOK specification. Solaris refers to the entire computer operating environment—the UNIX operating system, the OpenWindows window system, and the desktop environment—running on SPARC and Intel systems. SunSoft is the developer of Solaris and is a wholly-owned subsidiary of Sun Microsystems, Inc.

1.2 Solaris Desktop Integration Technologies

To overcome the limitations of the OPEN LOOK windowed user interface, and to move towards a fully integrated desktop, Solaris provides the following enhancements:

- An enhancement to the X Window System selections mechanism to provide a user-directed flow of information from one application to another (drag and drop)
- A method for applications to determine the identity and operating characteristics of objects on the desktop (the Classing Engine)
- A mechanism for passing messages and commands between applications (the ToolTalk service)

These enhancements represent three distinct technologies. Together these Solaris technologies provide the desktop with a powerful cohesiveness and data interchangeability between applications.

1.2.1 Drag and Drop

This mechanism allows users to use the mouse to exchange data between or within applications. This is done by selecting the graphical representation of the data with the Select and Adjust mouse buttons, clicking the representation (object) with the Select mouse button, keeping the mouse button down while

dragging the object to a destination, and dropping it (releasing the mouse button). The user need not be concerned with the subtleties of moving data between applications, such as the data's format, or whether data translation is required. All of this is handled by the application programmer using the drag and drop API.

Another application of Selection Services has been in use for some time. This involves using the mouse to select data, as outlined above, then using the copy and paste menu functions to move the data to another client.

Selection Services are covered in Chapter 2. The drag and drop API is presented later in this manual, starting with Chapter 3.

1.2.1.1 *Application*

Drag and drop can move data within an application. For example, a user can rearrange a text file by selecting text, dragging the text to a new position, and dropping (inserting) the text at that position.

Drag and drop can also move data between applications. An example of this would be copying data from Mail Tool and dropping it into a desktop publisher. Another example is copying an appointment from a mail message and dropping it onto Calendar Manager where it is properly entered.

1.2.1.2 *Drag and Drop and the X Server*

Selections and drag and drop provide a communications link between an *owner* client (the client that owns the data) and a *requestor* client (the client that receives the data). All data transferred through Selection Services and drag and drop is transferred through the X server. Each toolkit (OLIT or XView) provides a selections and drag and drop API. Though the APIs for each toolkit are somewhat different, selection between the toolkits is seamless and invisible.

1.2.2 *Classing Engine*

The Classing Engine is a database that identifies the characteristics, or *attributes*, of desktop objects. In other words, the Classing Engine stores attributes of desktop objects, such as print method, icons, and file opening commands.

If an application is to interoperate with other objects, the application must be able to identify those objects and determine their various operating characteristics. That is, if the object is another application, can they intercommunicate? If the object is a file, can the application read it? Every object on the desktop must be readily identifiable—is it an ASCII file, a SunSoft DeskSet tool, a spreadsheet program, a data file from a desktop publishing program, or something else? The Classing Engine provides a database for storing this information and an API to access the information. Applications query the Classing Engine database to determine an object's type and the attributes associated with the object. Chapter 5, "Classing Engine" discusses this technology in detail.

1.2.2.1 *Application*

File Manager, a DeskSet application shipped with OpenWindows, provides the best example of using the Classing Engine. File Manager graphically displays a UNIX file system as a set of folders (directories) and documents (files). Users can move, copy, and rearrange files by dragging and dropping file icons into directory icons. Files may be deleted by dropping icons into the waste basket icon. File Manager uses startup information stored in the Classing Engine to allow users to double click on a file icon and open the file with its associated application. For example, double clicking on a spreadsheet data file icon opens the file into the spreadsheet program. Double clicking a desktop publishing file opens the file into the correct desktop publishing application. Using print instructions from the Classing Engine, File Manager also allows users to print a data file by dragging and dropping it on the Print Tool.

Another feature of File Manager is that different file types are represented by different icons. Thus, one application's file will have one type of icon, and another application will have a different icon. Again, the Classing Engine provides File Manager with the icon display information.

1.2.3 *ToolTalk Service*

The ToolTalk service is used by applications to communicate with each other without having direct knowledge of each other. Applications communicate by creating and sending ToolTalk messages. The ToolTalk service receives these messages, determines the recipients, and then delivers the messages to the appropriate applications.

To use ToolTalk in your application, you must first decide what message protocol your application will implement. A message protocol is a set of ToolTalk messages that describe operations that applications agree to perform. By adopting a message protocol, applications can speak the same ToolTalk language. The message protocol specification includes the set of messages, as well as how applications will behave when they receive the messages. Refer to Chapter 6, "The ToolTalk Service" for further details on ToolTalk. Refer to Chapter 7, "The ToolTalk Service and DeskSet Integration" for a discussion on the DeskSet message protocol.

1.2.3.1 *Application Example*

Calendar Manager, another DeskSet application shipped with OpenWindows, provides an interesting example of how the ToolTalk service is used on the Solaris desktop. Calendar Manager (or CM) has the ability to browse multiple users' calendars, and schedule a meeting in a time slot that is open for all of them. CM needs to be able to send an electronic mail message to the relevant users informing them of the new meeting. There are several ways CM could do this.

One way would be for CM to open a text window to allow the user to enter the body of the mail message, after which CM would invoke the system's mail program to send the message. However, this would mean the CM has to implement its own facility for composing electronic mail messages — a facility not likely to be similar to the user's usual mail composition tool. The result is duplication of effort by developers and inconsistent facilities for users.

Another way would be for CM to create a temporary mail message file, look up the user's preferred mail composition tool, and invoke that tool on that file. The problem here is that communication between CM and the mail tool is limited to command line arguments and the process environment. (See the command `environ(5)`). For example,

- The mail tool has no standard way of finding out where on the screen it should appear.
- CM has no way of knowing if there already exists a running instance of the mail tool that could respond more quickly, or bring to bear on the job some state information that the user considers valuable.
- CM has no way of (for example) iconifying the composition window if the user wants to temporarily put away CM and its associated windows.

- CM has no way of gracefully aborting the composition operation if the user suddenly decides to quit.

The Solaris Desktop does not suffer from these constraints. In Solaris, CM can issue a ToolTalk request asking that the indicated mail message be edited. The ToolTalk service routes the request to the running mail editing tool that is best prepared to handle it, or starts the user's favorite mail tool if no instance is already running. The handling tool is placed in communication with the requesting CM, and is free to inquire about where on the screen to appear or what internationalization locale to assume, and to send back periodic status reports or data checkpoints. In turn, CM is able to monitor and control the operation, and manipulate the composition window almost as if it were one of its own. The result is that the user concentrates on his work, and not on his tools.

1.3 ISV Registration

Maximum desktop integration requires public notice of application data types, naming conventions, custom icon design and ToolTalk message protocols. By making this information public, ISVs can be sure that their applications and data files are recognized by other applications. SunSoft provides a vehicle for making this information public through the Developer Integration Format Registration program. Independent software vendors can register the data type information for their applications with SunSoft. This information will be made available to other ISVs through SunSoft. Refer to Appendix C, "Vendor Data Type Registration" for details.

1