# Glossary of Copland Terminology

*The terms defined in this glossary are preliminary and subject to change.*

**action** The objects and activities that make a human interface task unique, such as the names of folders to back up or a list of people to whom a message should be sent. See also **task**.

**activation model** The tasking and synchronization model mandated by an I/O family. It consists of the code that provides the runtime environment to the family and its plug-ins. See **family, plug-in.**

**active assistance** General term for Copland features that allow the computer to assist the user actively. See also **automation, delegation, task, task definition, assistant.**

**active window** The frontmost window on the desktop; the window where the next action will take place. The active window is identified by distinctive details that aren't visible for inactive windows.

**address space** The set of addresses that a process can reference.

**alert box** An arrangement of panels within a modal window that an application displays on the screen to warn the user or to report an error. An alert box typically consists of text describing the situation and buttons that require the user to acknowledge or rectify the problem. An alert box may or may not be accompanied by an alert sound. See also **dialog box, modal window.**

**area** A logical extent of memory with common attributes. Areas never overlap; a particular address in an address space is included in at most one area. See also **global area.**

**assistant** An entity that provides a specific kind of active assistance in a given context by asking the user questions and then taking actions or creating human interface tasks based on the answers.

**automation** Automatically controlled operations. In Copland, features that allow the computer to create a series of actions, or a human interface task, in a form that can be repeated. See also **task.**

**backing provider** A server process responsible for transferring memory between backing storage (for example, a hard disk) and physical memory in response to page faults.

**bus expert** See **low-level expert**.

**cell** A rectangular part of a list displaying information about one item in the list.

**CFM-based software** Software compiled for execution in a runtime environment that uses Code Fragment Manager (CFM) fragments to organize executable code and data in memory.

**close box** The box on the left end of the title bar of an active window. Clicking it closes the window.

**code fragment** See **fragment.**

**Code Fragment Manager** The part of the Mac OS that loads fragments into memory and prepares them for execution.

**collapse box** The box on the right end of the title bar of an active window. Clicking it collapses or expands the window.

**condition** The set of events or states that trigger a human interface task. See also **task.**

**cooperative process** A process that has a cooperative task created by the Process Manager. The act of relinquishing control of the processor at a well-defined time, such as at a call to the `WaitNextEvent` function, is a defining characteristic of a cooperative process. See **process, server process.**

**cooperative task** A task created by the Process Manager for an application when the application is launched. Scheduling of cooperative tasks is layered on top of preemptive scheduling of preemptive tasks. The microkernel sees all cooperative and preemptive tasks as peers and schedules them preemptively. The Process Manager ensures that for all cooperative processes, only one cooperative task is eligible to run at any one time. See also **preemptive task, task.**

**cooperative thread** See **thread.**

**data-driven patching** A method of patching according to which you create data structures specifying the address of your patch and of the patched routine, the name of your patch, and any other information required to execute your patch. Using this model, the task of installing a patch and scheduling patch execution is taken over by the Code Fragment Manager and the Patch Manager. This is the preferred patching method in Copland.

**data fork** Part of a file that contains data accessed using the File Manager. The data usually corresponds to data entered by the user; the application creating a file can store and interpret the data in the data fork in whatever manner is appropriate.

**delegation** General term for Copland features that allow the computer to trigger human interface tasks when a specified condition occurs. See also **condition, task.**

**desktop animation module** A module controlled by the Desktop Animation Manager that can maintain the appearance of the desktop background or act as a screen saver when no keyboard or mouse events occur within a specified period of time.

**Device Manager family** A Copland I/O family to which two types of device drivers belong: those that have not been revised to run as a plug-in to another I/O family and those for which no family exists. See **family, plug-in.**

**dialog box** An arrangement of panels within a window that an application displays to solicit information from the user before the application carries out the user's command. See also **modal dialog box, modeless dialog box, movable modal dialog box.**

**DLL** See **dynamically linked library.**

**document information panel** A composite panel, available via the Document Info command in an application's File menu, that presents the same information presented by the Get Info command in the Finder's File menu. The document information panel also appears in an application's Save and Open dialog boxes. See also **panel**.

**document window** A window of class `kDocumentWindowClass`, typically used to display document data. A document window appears behind floating and modal windows in an application's layer. See also **floating window, modal window.**

**drag** To position the pointer on a visual interface element (such as an icon), press and hold the mouse button, move the pointer to a new position, and then release the mouse button. Dragging can have different effects, depending on what's under the pointer when the user first presses the mouse button. To support the dragging of items from one place to another, you use the Drag Manager.

**Driver Loader Library** A set of routines that all I/O families can use to locate and instantiate their plug-ins.

**Driver Services Library** A set of routines that provide basic driver services to families, such as interrupt registration, timing facilities, allocation and deallocation of memory, and secondary-interrupt-handling capabilities.

**Dynamic Host Configuration Protocol (DHCP)** An Internet standard for managing end-node Internet Protocol (IP) configuration. DHCP automates the configuration process by returning information about the TCP/IP network to the user's configuration control panel.

**dynamically linked library (DLL)** A shared library that is automatically loaded by the Code Fragment Manager at runtime in order to export code or data referenced by another fragment.

**embedding panel** A panel that contains other panels and controls their layout, keyboard focus, and mouse interactions.

**entry ID** An opaque reference assigned by the Name Registry to a registry entry. It uniquely identifies a registry entry until the system restarts. If the registry entry is deleted, the Name Registry may reuse the entry ID after a long period of time.

**entry name** A character string that characterizes a registry entry in the Name Registry. It may name a general category of things or a specific thing. For instance, the entry name of the registry entry representing the device tree is "devices". The entry name of a registry entry representing a specific device might be "ADB keyboard".

**exception** An error or other special condition detected by the microprocessor in the course of program execution.

**execution environment** A set of conventions regarding how code gets activated and what services and memory

are available to it. See also **hardware interrupt level, secondary interrupt level, software interrupt level, task level.**

**expert** See **family expert, low-level expert.**

**extent** Continuous memory space reserved on backing storage or in physical RAM for data or code.

**factoring** Using Apple events to separate the code that controls an application's user interface from the code that responds to the user's manipulation of the interface. In a fully factored application, any significant user actions generate Apple events that any scripting component based on the Open Scripting Architecture (OSA) can record as statements in a compiled script.

**family** (1) A collection of software pieces that provide a single set of I/O services to the system, such as the SCSI family and its SCSI interface modules (SIMs) or the file systems family and its installable file systems. Often, a family is associated with a set of devices that have similar characteristics, such as display devices or ADB devices. (2) A collection of devices that provide the same kind of I/O services, for example, the family of display devices.

**family expert** Code within a family that maintains knowledge of the set of family plug-ins within the system. It locates family plug-ins in the Name Registry and instantiates them. Sometimes referred to as a *high-level expert*.

**family programming interface (FPI)** An I/O family's API that provides applications, other families, and system software with access to the family's services.

**family server** See **FPI server.**

**family services library** A set of services to which a family's plug-ins can subscribe. Such services can include communicating data, allocating memory, and registering and servicing interrupts. It provides services that supplement those available from the microkernel.

**file** A named, ordered collection of information stored on a Mac OS volume, typically divided into a data fork and a resource fork.

**floating window** A window of class `kFloatingWindowClass`, typically used for tool palettes, catalogs, and other elements used to act on data in document windows. A floating window appears in front of document windows and behind modal windows in an application's layer. See also **document window, modal window.**

**FPI** See **family programming interface.**

**FPI server** Family software that runs in supervisor mode and responds to service requests from family clients.

**fragment** In Copland, the basic unit of executable code and its data.

**generic driver** A driver whose services are available through the Device Manager. Generic drivers are plug-ins to the Device Manager family. See **plug-in.**

**global area** An area mapped to all address spaces. Specifying this area attribute allows software residing in different address spaces to share data and code.

**global family constants** A set of shared family values.

**global patch** A patch that is called by every application referencing the patched routine. To create a global patch, you create a shared library that calls the patch and a special shared library fragment that contains the patch description structure for that patch. The shared library must use per-context instantiation.

**guard pages** Inaccessible pages of memory placed immediately before and after the range of addresses specified by an area. Copland does not actually allocate backing storage or RAM for guard pages; it merely marks the addresses in those pages as inaccessible to user-level or supervisor-level software.

**hardware interrupt** An exception signaled to the processor by a physical device, notifying the processor of a change of condition of the device, such as the reception of incoming data.

**hardware interrupt handler** Code that is invoked as a direct result of a hardware interrupt. A hardware interrupt handler always runs in supervisor mode.

**hardware interrupt level** The execution environment in which a hardware interrupt handler runs. Only a subset of microkernel and OS services are available. No Toolbox services are available. Only memory that is physically resident is accessible; page faults at hardware interrupt level are illegal and system fatal. See **page fault.**

**head patch** A patch that does some processing before it calls the patched routine. See also **surround patch** and **tail patch**.

**high-level expert** See **family expert.**

**interface definition objects (IDOs)** Copland's SOM-based replacements for the definition procedures (defprocs) used in System 7 to draw windows, menus, and controls.

**interrupt** See **hardware interrupt, secondary interrupt, software interrupt.**

**interrupt handler** A routine that services interrupts. See also **hardware interrupt handler, secondary interrupt handler, software interrupt handler.**

**interrupt latency** The time between the generation of an interrupt and the execution of its associated interrupt handler.

**kernel** See **microkernel**.

**keyboard focus** The focal point on screen for actions triggered by keypresses.

**list** A series of items displayed within a rectangle. Lists may have zero, one, or two scroll bars.

**local patch** A patch that executes only within your application's context. You create a local patch by including a patch description structure for it in a special fragment associated with your application fragment.

**low-level expert** Code that has specific knowledge of a piece of hardware, such as a bus or a main logic board. It knows how physical devices are connected to the

system, and it installs and removes that information in the device portion of the Name Registry. Sometimes referred to as a *motherboard expert*.

**memory area** A range of addresses, within an address space, sharing common attributes.

**microkernel** The set of lowest-level operating system services, including memory management, task management, synchronization primitives, interprocess communication mechanisms, interrupt handling, and basic timing services.

**modal dialog box** A dialog box displayed in a modal window that the user can't move. A modal dialog box resembles an alert box. The user can dismiss a modal dialog box only by clicking its buttons. See also **modeless dialog box, movable modal dialog box.**

**modal window** A window of class `kModalWindowClass` that puts the user in a state or "mode" of being able to work only inside the window. A modal window is typically used to display a dialog box or alert box that requires immediate attention from the user. A modal window appears in front of all other windows in an application's layer. See also **document window, floating window.**

**modeless dialog box** A dialog box displayed in a document window without a size box or scroll bars. The user can move a modeless dialog box, make it inactive and active again, and close it like any document window. See also **modal dialog box, movable modal dialog box.**

**modifier** Information associated with a registry entry name or property that is hardware or implementation specific, such as whether or not the name or property is saved to nonvolatile RAM.

**motherboard expert** See **low-level expert.**

**movable modal dialog box** A dialog box displayed in a modal window with a title bar (but no close box) that the user can drag to move the box. The user can dismiss a movable modal dialog box only by clicking its buttons. See also **dialog box, modal dialog box, modeless dialog box.**

**multicast** A message that is received by any number of hosts that are registered members of a group.

**multihoming** A networking feature that allows multiple network interface controller (NIC) cards to be active on a single node at the same time.

**multinode architecture** An AppleTalk feature that allows an application to acquire node IDs that are additional to the standard node ID assigned to the system when the node joins an AppleTalk network. Multinode architecture is provided to meet the needs of special-purpose applications that receive and process AppleTalk packets in a custom manner, instead of passing them directly on to a higher-level AppleTalk protocol for processing. A multinode ID allows the system running your application to appear as multiple nodes on the network.

**name entry** See **registry entry**.

**Name Registry** A high-level Mac OS system service that stores the names of software objects and the relations among

the names. For instance, the Name Registry contains device information and makes it available to Macintosh run-time drivers.

**network-layer protocol** The protocol level directly above the data-link layer. The network layer is responsible for routing data between systems on the network.

**node** A device addressable at the network-layer protocol level.

**nonprivileged software** Code that executes while the processor is in user mode. To protect the state of the user's system, applications should typically execute as user-level software.

**non-reentrant code** Code that should be executed by only one piece of software at a time.

**notification** The way in which a human interface task notifies the user that a task has been completed; for example, a notification might write details to a log and page the user via a commercial paging system when a task is completed. See also **task**.

**object** A file system container that holds properties. Certain types of file system objects can contain other file system objects. File system objects include files, folders, volumes, and the universe.

**Open Transport** A communications architecture that can be used to implement any number of networking and other communications systems. It replaces the AppleTalk, MacTCP, and Serial Driver interfaces with a single, transport-independent interface.

**page fault** An exception that causes data to be transferred between backing storage (for example, a hard disk) and physical memory.

**panel** (1) A SOM object that encapsulates one or more human interface elements. (2) A hierarchy of embedded panels that make up a larger control panel, document information panel, or other integrated set of human interface elements. See also **document information panel**.

**patch** A piece of code that intercepts the transaction between a client and a service that is implemented by a single routine. You can assume control in order to monitor the use of that service, to modify the service, or to replace the service. A patch is uniquely identified by a patch ID.

**patch chain** An ordered list of patches on the same instance of the same entry point. A patch chain is uniquely identified by a patch chain ID.

**patch chain ID** A unique identifier that specifies a patch chain. The Patch Manager uses this identifier to distinguish among patch chains belonging to the same patch chain set.

**patch chain set** The set of patch chains associated with a single context.

**patch description structure** A data structure you create to describe a patch. It specifies the address of the patch, the address of the patched routine, the name of the patch, and options related to the execution of the patch. You store patch description structures in a special fragment

that is associated either with the application (for local patching) or a shared library (for global patching).

**patch ID** A unique identifier that specifies a patch in a patch chain. The Patch Manager returns this ID to you after it installs a patch.

**plug-in** (1) A dynamically loaded piece of software that provides an instance of the service provided by a family. Within the file systems family, for example, a volume-format plug-in implements file-system services for a specific volume format. (2) A dynamically loaded piece of software that provides a service for an application.

**plug-in programming interface (PPI)** A family–to–plug-in interface that defines the entry points a plug-in must support so that it can be called and a plug-in–to–family interface that defines the routines plug-ins must call when certain events occur. In addition, a PPI defines the data request path through which the family and its plug-ins exchange data.

**pool** A portion of logical memory from which software can dynamically allocate memory. A pool is contained in one or more areas.

**Pool Manager** In Copland, a reentrant, pointer-based heap allocator that allows software to dynamically allocate memory without affecting other areas of memory.

**PowerTalk system software** Apple Computer's implementation of the AOCE system software for use on Mac-compatible computers. The PowerTalk system software includes desktop services, as well as Mac OS managers and utility functions that provide APIs for catalog, messaging, and security services.

**preemptive task** A task that uses only reentrant operating-system services, allowing it to be preemptively scheduled by the microkernel. See also **cooperative task, task.**

**primary human interface** The menus, windows, and tools that allow a user to control an application directly. See also **secondary human interface.**

**primary interrupt level** See **hardware interrupt level.**

**primary task** See **cooperative task.**

**priority** The ranking of a task used by the Copland microkernel for execution scheduling. The microkernel provides various symbolic priorities for applications, drivers, servers, and real-time operations.

**privilege level** An access state to processor resources corresponding to the mode of the processor. Code that has the privilege to execute while the processor is in supervisor mode is called *privileged software,* and code that has the privilege to execute only while the processor is in user mode is called *nonprivileged software*.

**privileged software** Code that executes while the processor is in supervisor mode. Typically, only the operating system and portions of device drivers should run in supervisor mode in Copland.

**process** A set of one or more tasks and the memory and other operating system resources allocated to those tasks. For

example, when Copland launches an application, the system creates a process that identifies the application's cooperative task, any preemptive tasks created by the application, and the memory areas allocated to those tasks. A process is specific to one address space, but multiple processes can share the same address space. Copland uses processes to track resource allocation and to reclaim resources. See also **cooperative process, server process.**

**Process Manager** In Copland, the part of Mac OS that manages the scheduling of cooperative processes and that controls access to resources shared by those processes.

**programmatic patching** A method of patching used with System 7 according to which you replace the address of an existing operating system routine in the trap dispatch table with the address of a patch routine. This method is supported in Copland but not in any Mac OS release following that.

**promise** A scrap that contains placeholders for scrap item types rather than the scrap item data itself. When the user pastes or completes a drag, the Scrap Manager asks the original application to fulfill its promise for the type of data being pasted or dropped and provide the actual data. See also **scrap.**

**property** (1) A unit of data belonging to a registry entry. A property stores information about the thing represented by the registry entry. Each property consists of a property name and and a property value. (2) A data item or a set of data that is stored by the file system.

**property name** A null-terminated character string that characterizes a property of a registry entry. Examples of property names include "vendor-id" and "driver-description".

**property value** The data in a property.

**reentrant services** Code written so that the data it manipulates is kept logically separate from the code itself, allowing the code to be safely called by several pieces of software at the same time.

**registry entry** A node in the Name Registry. A registry entry is connected to one or more other entries by relationships, such as parent, child, and sibling. Each registry entry has an entry name, an entry ID and one or more properties.

**replacement patch** A patch that never calls the routine it is patching; it replaces it completely.

**resource** Any data stored according to a defined structure in a resource fork of a file. The data in a resource is interpreted according to its resource type.

**resource fork** Part of a file that contains the file's resources.

**resource ID** A number that identifies a specific resource of a given resource type.

**resource type** A sequence of four characters that uniquely identifies a specific type of resource.

**routine descriptor** A data structure that indicates the instruction set architecture of a particular routine by describing the routine's address, its parameters, and its calling conventions.

**runtime environment** The set of conventions that determine how code is loaded into memory, where data is stored and how it is addressed, and how functions call other functions and system software routines.

**scrap** A structure created by the Scrap Manager that consists of one or more scrap items that can hold one ore more pieces of data. See also **promise, scrap item, scrap item type.**

**scrap item** A structure created by the Scrap Manager for holding a single piece of data that can be represented in different ways by one or more scrap item types. See also **scrap, scrap item type.**

**scrap item type** A structure created by the Scrap Manager that is associated with a scrap item and holds a single representation of the piece of data associated with the scrap item. See also **scrap, scrap item**.

**secondary human interface** A high-level interface that frees users from detailed decisions about how to make the computer perform specific actions. For example, assistants make decisions on the user's behalf and use the primary human interface to carry them out. See also **assistant**, **primary human interface**.

**secondary interrupt** A signal sent to the microkernel by a primary interrupt handler or a privileged task requesting that a secondary interrupt handler be queued for execution.

**secondary interrupt handler** A routine that runs as a result of a secondary interrupt. A secondary interrupt handler always runs in supervisor mode. A secondary interrupt handler can be preempted only by hardware interrupt handlers.

**secondary interrupt level** The execution environment in which a secondary interrupt handler runs. Only a subset of microkernel and OS services are available. No Toolbox services are available. Only memory that is physically resident is accessible; page faults at secondary interrupt level are illegal and system fatal.

**secondary task** See **preemptive task.**

**semaphores** Synchronization primitives used to block (schedule or switch out) tasks until a required resource becomes available. For example, a global semaphore could be used to restrict access to the Toolbox to one application at a time.

**server** (1) A computer and associated software that provide a service to users and that control access to that service, such as a file server or a database server. (2) In Copland, a process that provides a service to other processes on the same or a connected computer. See also **server process.** (3) In the context of an I/O family, software built on the client/server model, but that has a single client. See also **FPI server.**

**server process** A process that has its own address space and whose task (or tasks) uses only reentrant system services. See also **cooperative process.**

**shared library** (1) A fragment that exports functions and variables to other fragments. A shared library is used to resolve imported symbols during linking and also during the loading and preparation of some other fragment. Some shared libraries are dynamically linked; others must be explicitly instantiated during execution. (2) Any fragment.

**size box** A box in the lower-right corner of some active windows. Dragging the size box resizes the window.

**software interrupt** A signal sent to the microkernel requesting that it invoke a particular routine that, upon completion, returns execution at the point where the interrupt was sent.

**software interrupt handler** A routine that runs in a particular task as a result of a software interrupt. A software interrupt handler can be preempted by other tasks, secondary interrupt handlers, and hardware interrupt handlers. Software interrupt handlers can access virtual memory.

**software interrupt level** The execution environment in which a software interrupt handler runs. All microkernel, OS, and Toolbox services available at task level are available at software interrupt level. See also **task level.**

**SOM** See **System Object Model.**

**supervisor mode** A state of operation for the PowerPC processor that allows software to gain access to all of memory, all processor registers, and other critical resources. Only software with supervisor-mode privilege can switch the processor between supervisor mode and user mode. Compare **user mode.**

**supervisor-mode software** See **privileged software.**

**surround patch** A patch that performs some processing, calls the patched routine, and then performs some additional processing. This is the model used for all types of data-driven patching. See also **head patch** and **tail patch.**

**System Object Model (SOM)** A technology from International Business Machines, Inc., that provides language-independent and platform-independent means of defining programmatic objects and handling method dispatching dynamically at runtime.

**tail patch** A patch that does some processing after calling the patched routine. See also **head patch** and **surround patch.**

**task** (1) In the Copland human interface, a persistent representation of a sequence of actions that can be triggered programmatically. A task is created from a task definition in a manner analogous to the way a document is created from an application. See also **task definition**. (2) In Copland system software, the basic unit of program execution that is preemptively scheduled by the microkernel. A task has its own stack and set of registers and it may share the same address space with other

tasks. A task executes in either user mode or supervisor mode. In Copland, System 7 tasks such as Time Manager tasks and deferred tasks are invisible to the microkernel. See also **cooperative task, preemptive task.**

**task definition** A definition of how a particular kind of human interface task is to be performed. Combined with information about the parameters for a specific task, such as filenames or other details and a condition, a task definition can be used by an assistant or directly by a user to create one or more tasks. See also **task**.

**task level** The execution environment in which all tasks run. Most microkernel, OS, and Toolbox services are available at task level. Task level software can access virtual memory.

**task switch** The act of suspending one task's execution and resuming a different task's execution. In a task switch, the microkernel saves the processor state for the suspended task and restores the processor state of the task resuming execution. The microkernel performs a task switch based on the priority of tasks that are eligible to execute and its time-slicing mechanism. See also **priority, time slice.**

**text object** A private data structure that contains information about both text content and text encoding and that takes the place of both Pascal and C strings.

**theme** A coordinated set of human interface designs that determine the appearance of human interface elements on a systemwide basis.

**thread** Within a task, a sequence of instructions and the processor context to execute it, including a register set, a program counter, and a stack. In Copland, all threads are cooperatively scheduled by the Thread Manager.

**time slice** A defined interval of time during which a task is allowed to execute.

**title bar icon** An icon in a document window's title bar that the user can drag to a new volume to copy the document to that volume.

**transport independence** A property of a communications architecture that makes the set and sequence of functions called by an application independent of the underlying network protocols used to transmit or receive data. The set of functions called and the sequence of calls depends solely on the nature of the communication, not on the protocol or protocol family used.

**trusted software** Code that executes in supervisor mode and has access to the microkernel's protected memory space. See **privileged software.**

**universe** The file system object that contains all the volumes mounted on a given computer system. The universe can also include file system properties, files, and folders.

**user mode** A state of operation for the PowerPC processor that allows software, typically application software, to execute in an environment that protects certain critical resources, such as portions of memory and certain processor registers. Compare **supervisor mode.**

**user-mode software** See **nonprivileged software.**

**VBL** See **vertical retrace interrupt**.

**vertical blanking interrupt (VBL)** See **vertical retrace interrupt.**

**vertical retrace interrupt** An interrupt generated by the video circuitry each time the electron beam of a monitor's display tube returns from the lower-right corner of the screen to the upper-left corner.

**workspace** One of several separate custom user environments for a single computer.

**zoom box** A box to the left of the collapse box in a window's title bar that the user can click to alternate between two different window sizes. Clicking the zoom box once causes the window to expand so it fills the screen of the monitor on which it is displayed. Clicking the zoom box a second time restores the window to its previous size and location. See also **collapse box.**