# Glossary

***abstract superclass*** A superclass listed in the *Apple Event Registry: Standard Suites*, such as `cObject` or `cOpenableObject`, that is used only in definitions of object classes and not for real Apple event objects. See also ***object class.***

***active function*** A function called by a scripting component periodically during script compilation and execution. You must provide an alternative active function for the use of scripting components if you want your application to get time during script compilation and execution for tasks such as spinning the cursor or checking for system-level errors.

***additional parameter*** A keyword-specified descriptor record that a server application uses in addition to the data specified in the direct parameter. For example, an Apple event for arithmetic operations may include additional parameters that specify operands in an equation. Additional parameters may be required, or they may be optional.

***address descriptor record*** A descriptor record of data type `AEAddressDesc` that contains the address of the target or source of an Apple event.

***AEIMP*** See ***Apple Event Interprocess Messaging Protocol.***

***AE record*** A descriptor record of data type `AERecord` that usually contains a list of parameters for an Apple event. See also ***Apple event parameter.***

***Apple event*** A high-level event that adheres to the Apple Event Interprocess Messaging Protocol. An Apple event consists of attributes (including the event class and event ID, which identify the event and its task) and, usually, parameters (which contain data used by the target application for the event). See also ***Apple event attribute, Apple event parameter.***

***Apple event array*** An array in a descriptor list. The data for an Apple event array is specified by an array data record, which is defined by the data type `AEArrayData`.

***Apple event attribute*** A keyword-specified descriptor record that identifies the event class, event ID, target application, or some other characteristic of an Apple event. Taken together, the attributes of an Apple event identify the event and denote the task to be performed on the data specified in the Apple event's parameters. Unlike Apple event parameters (which contain data used only by the target application of the Apple event), Apple event attributes contain information that can be used by both the Apple Event Manager and the target application. See also ***Apple event parameter.***

***Apple event dispatch table*** A table in either the application heap or the system heap that the Apple Event Manager uses to map Apple events to the appropriate Apple event handlers.

***Apple event handler*** An application-defined function that extracts pertinent data from an Apple event, performs the action requested by the Apple event, and returns a result.

***Apple Event Interprocess Messaging Protocol (AEIMP)*** A standard defined by Apple Computer, Inc., for communication and data sharing among applications. High-level events that adhere to this protocol are called Apple events. See also ***Apple event.***

***Apple Event Manager*** The collection of routines that allows client applications to send Apple events to server applications for the purpose of requesting services or information.

***Apple event object*** A distinct item in a target application or any of its documents that can be specified by an object specifier record in an Apple event sent by a source application. Apple event objects can be anything that an application can locate on the basis of such a description, including items that a user can differentiate and manipulate while using an application, such as words, paragraphs, shapes, windows, or style formats. See also ***object specifier record.***

***Apple event object class*** See ***object class.***

***Apple event parameter*** A keyword-specified descriptor record containing data that the target application for an Apple event uses. Unlike Apple event attributes (which contain information that can be used by both the Apple Event Manager and the target application), Apple event parameters contain data used only by the target application of the Apple event. See also ***Apple event attribute, direct parameter, optional parameter, required parameter.***

***Apple event record*** A descriptor record of data type `AppleEvent` that contains a list of keyword-specified descriptor records. These descriptor records describe—at least—the attributes necessary for an Apple event; they may also describe parameters for the Apple event. Apple Event Manager functions are used to add parameters to an Apple event record.

***Apple event user terminology resources*** Two resources with identical formats used by server applications to specify the Apple events and corresponding user terminology that the applications support. The `'aeut'` resource, which is provided by scripting components, contains terminology information for all the standard suites of Apple events defined in the *Apple Event Registry: Standard Suites*. An `'aete'` resource must be provided by every scriptable application; it describes which of the standard suites listed in the `'aeut'` resource the application supports and provides additional terminology information for extensions to the standard suites and custom Apple events supported by the application. See also ***scripting component.***

***AppleScript component*** The scripting component that implements the AppleScript scripting language. See also ***scripting component.***

***AppleScript scripting language*** The standard user scripting language defined by Apple Computer, Inc. The AppleScript scripting language is implemented by the AppleScript scripting component. See also ***dialect.***

***application result handler*** A result handler that is associated with a particular application. Compare with ***system result handler.***

***asynchronous parameter block*** In the Data Access Manager, the parameter block that allows a routine to return control to your application before the routine has completed execution.

***authentication*** The process of establishing the identity of a user. The authentication mechanism of the PPC Toolbox identifies each user through an assigned name and password.

***boundary objects*** The elements, specified in a range descriptor record, that identify the beginning and end of the range. See also ***range descriptor record.***

***client application*** An application that uses Apple events to request a service (for example, printing a list of files, checking the spelling of a list of words, or performing a numeric calculation) from another application (called a server application). These applications can reside on the same local computer or on remote computers connected to a network.

***coercion handler*** A routine that coerces data from one descriptor type to another.

***coercion handler dispatch table*** A table in either the application heap or the system heap that the Apple Event Manager uses to map desired coercions to the appropriate coercion handler. See also ***coercion handler.***

***comparison descriptor record*** A coerced AE record of type `typeCompDescriptor` that specifies an Apple event object and either another Apple event object or data for the Apple Event Manager to compare to the first object.

***compiled script*** Compiled code that a client application can decompile into source data or execute using the standard scripting component routines.

***compiled script file*** A script file with the file type `'scpt'` that contains script data as a resource of type `'scpt'`. Before executing the script in a compiled script file, a user must first open the script from the Finder or from an application such as Script Editor.

***component-specific storage descriptor record*** A descriptor record returned by `OSAStore`. The descriptor type for a component-specific storage descriptor record is the scripting component subtype value for the scripting component that created the script data.

***container*** An Apple event object that contains another Apple event object. A container is specified in an object specifier record by a keyword-specified descriptor record with the keyword `keyAEContainer`. The keyword-specified descriptor record is usually another object specifier record. It can also be a null descriptor record, or it can be used much like a variable when the Apple Event Manager determines a range or performs a series of tests. The objects a container contains can be either elements or properties. See also ***Apple event object, element, object specifier record, property.***

***container hierarchy*** The chain of containers that determine the location of one or more Apple event objects. See also ***container.***

***core Apple event*** An Apple event defined as part of the Core suite of Apple events in the *Apple Event Registry: Standard Suites.*

***create function*** A function called by a scripting component whenever it creates an Apple event during script execution. You must provide an alternative create function if you want to gain control over the creation and addressing of Apple events. If you don't provide an alternative create function, scripting components call the standard Apple Event Manager function `AECreateAppleEvent` with default parameters.

***custom Apple event*** An Apple event you define for use by your own applications. Instead of creating custom Apple events, you should try to use the standard Apple events and extend their definitions as necessary for your application. If you think you need to define custom Apple events, you should check with the Apple Event Registrar to find out whether Apple events that already exist or are under development can be adapted to the needs of your application.

***database extension*** The interface between the Data Access Manager and a data server.

***data server*** An application that acts as an interface between a database extension on a Macintosh computer and a data source, which can be on the Macintosh computer or on a remote host computer. A data server can be a database server program that can provide an interface to a variety of different databases, or it can be the data source itself, such as a Macintosh application.

***default container*** The outermost container in an application's container hierarchy; usually the application itself. See also ***container hierarchy.***

***default scripting component*** The scripting component used by the generic scripting component when an application passes `kOSANullScript` rather than a valid script ID to `OSACompile` or `OSAStartRecording`.

***descriptor list*** A descriptor record of data type `AEDescList` whose data handle refers to a list of descriptor records.

***descriptor record*** A data structure of type `AEDesc` that consists of a handle to data and a descriptor type that identifies the type of the data referred to by the handle. Descriptor records are the fundamental data structures from which Apple events are constructed.

***descriptor type*** An identifier for the type of data referred to by the handle in a descriptor record.

***dialect*** A version of a scripting language that resembles a specific human language or programming language; for example, the AppleScript scripting language provides dialects that resemble English, Japanese, and other languages. See also ***AppleScript scripting language.***

***direct parameter*** The parameter in an Apple event that contains the data or object specifier record to be used by the server application. For example, a list of documents to be opened is specified in the direct parameter of the Open Documents event. See also ***Apple event parameter.***

***edition*** The data written to an edition container by a publisher. A publisher writes data to an edition whenever a user saves a document that contains a publisher, and subscribers in other documents may read the data from the edition whenever it is updated. See also ***publisher, subscriber.***

***edition container*** A file that holds edition data, represented on the desktop by an edition icon. An edition container obtains its data from a publisher within a document. See also ***edition, publisher.***

***Edition Manager*** The collection of routines that allows applications to automate copy and paste operations between applications, so that data can be shared dynamically.

***element*** An Apple event object contained by another Apple event object specified as the element's container. An Apple event object can contain many elements of the same element class, whereas an Apple event object can have only one of each of its properties. See also ***Apple event object, container, element classes, property.***

***element classes*** In the *Apple Event Registry: Standard Suites,* a list of the object classes for the elements that an Apple event object of a given object class can contain. See also ***Apple event object, object class.***

***error callback function*** An object callback function that gives the Apple Event Manager an address. The Apple Event Manager writes to this address the descriptor record it is currently working with if an error occurs during the resolution of an object specifier record. See also ***object callback function.***

***event class*** An attribute that identifies a group of related Apple events. The event class appears in the `message` field of the Apple event's event record. The event class and the event ID identify the action an Apple event performs. See also ***Apple event attribute, event ID.***

***event ID*** An attribute that identifies a particular Apple event within a group of related Apple events. The event ID appears in the `where` field of the Apple event's event record. The event ID and the event class identify the action an Apple event performs. See also ***Apple event attribute, event class.***

***Event Manager*** The collection of routines that an application can use to receive information about actions performed by the user, to receive notice of changes in the processing status of the application, and to communicate with other applications.

***extension*** An object class that duplicates all the characteristics of an object class of the same name and adds some of its own. Like a word in a dictionary, a single object class ID can have several related definitions.

***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 a scripting component can record as statements in a compiled script. See also ***recordable application.***

***functional-area Apple event*** A standard Apple event supported by applications with related features; for example, an Apple event related to text manipulation for word-processing applications, or an Apple event related to graphics manipulation for drawing applications. Functional-area Apple events are defined by Apple Computer, Inc., in consultation with interested developers and are published in the *Apple Event Registry: Standard Suites.*

***generic script ID*** Special script IDs used by the generic scripting component to keep track of script IDs provided by multiple scripting components. The generic scripting component translates generic scripting IDs into the corresponding component-specific script IDs and vice versa when necessary.

***generic scripting component*** A special scripting component that establishes connections dynamically with the appropriate scripting component for each script that a client application attempts to manipulate or execute. The generic scripting component also provides routines that you can use to determine which scripting component created a particular script, get an instance of a specific scripting component, and perform other useful tasks when you are using multiple scripting components. See also ***scripting component.***

***generic storage descriptor record*** A descriptor record of type `kOSAGenericStorage` that can be used by the generic scripting component or any other scripting component to store script data. The script data in a generic storage descriptor record is followed by a trailer that contains the subtype for the scripting component that created the script data.

***implied length*** The definition of a specific length for a data type. An example of this is the Data Access Manager's `typeInteger` data type, which has a defined length of 4 bytes.

***insertion location descriptor record*** A record of type `typeInsertionLoc` that consists of two keyword-specified descriptor records. The first is an object specifier record, and the data for the second is a constant that specifies the insertion location in relation to the Apple event object described by the object specifier record.

***interapplication communication (IAC) architecture*** A standard and extensible mechanism for communication among Macintosh applications, including the Edition Manager, the Open Scripting Architecture, the Apple Event Manager, the Event Manager, and the PPC Toolbox.

***key data*** The data in an object specifier record that distinguishes one or more Apple event objects from other Apple event objects of the same object class in the same container. Key data is specified by a keyword-specified descriptor record with the keyword `keyAEKeyData`. The Apple Event Manager interprets key data according to the key form specified in the same object specifier record.

***key form*** The form taken by the key data in an object specifier record. The key form is specified by a keyword-specified descriptor record with the keyword `keyAEKeyForm`. The keyword-specified descriptor record contains a constant that determines how the Apple Event Manager and a target application use the key data to locate specific Apple event objects. For example, the key form constant `formName` indicates that the key data consists of a name, which should be compared to the names of Apple event objects in the container specified by the object specifier record.

***keyword*** A four-character code that uniquely identifies a descriptor record inside another descriptor record. In Apple Event Manager functions, constants are typically used to represent the four-character codes.

***keyword-specified descriptor record*** A record of data type `AEKeyDesc` that consists of a keyword and a descriptor record. Keyword-specified descriptor records are used to describe the attributes and parameters of an Apple event.

***location name*** An identifier for the network location of the computer on which a port resides. The PPC Toolbox provides the location name. It contains an object string, a type string, and a zone. An application can specify an alias for its location name by modifying its type string. See also ***port.***

***logical descriptor record*** A coerced AE record of type `typeLogicalDescriptor` that specifies a logical expression—that is, an expression that the Apple Event Manager evaluates to either `TRUE` or `FALSE`. The logical expression is constructed from a logical operator (one of the Boolean operators `AND`, `OR`, or `NOT`) and a list of logical terms to which the operator is applied. Each logical term in the list can be either another logical descriptor record or a comparison descriptor record.

***mark-adjusting function*** A marking callback function that unmarks objects previously marked by a call to an application's marking function.

***mark count*** The number of times the Apple Event Manager has called the marking function for the current mark token. Applications that support marking callback functions should associate the mark count with each Apple event object they mark.

***marking callback functions*** Object callback functions that allow your application to use its own marking scheme rather than tokens when identifying large groups of Apple event objects. See also ***mark-adjusting function, mark token function, object callback function,*** and ***object-marking function.***

***mark token*** A token returned by a mark token function. A mark token identifies the way an application marks Apple event objects during the current sessions while resolving a single test. A mark token does not identify a specific Apple event object; rather, it allows an application that supports marking callback functions to associate a group of objects with a marked set.

***mark token function*** A marking callback function that returns a mark token.

***message block*** A byte stream that an open application uses to send data to and receive data from another open application (which can be located on the same computer or across a network). The PPC Toolbox delivers message blocks to an application in the same sequence in which they were sent.

***Name-Binding Protocol (NBP)*** An AppleTalk protocol that maintains a table containing the internet address and name of each entity in the node that is visible to other entities on the internet (that is, each entity that has registered a name with NBP).

***null descriptor record*** A descriptor record whose descriptor type is `typeNull` and whose data handle is `NIL`.

***object accessor dispatch table*** A table in either the application heap or the system heap that the Apple Event Manager uses to map descriptions of objects in an object specifier record to object accessor functions that can locate those objects.

***object accessor function*** An application-defined function that locates an Apple event object of a specified object class in a container identified by a token of a specified descriptor type.

***object callback function*** An application-defined function used by the Apple Event Manager to resolve object specifier records. See also ***error callback function, marking callback functions, object-comparison function, object-counting function,*** and ***token disposal function.***

***object class*** A category for Apple event objects that share specific characteristics listed in an object class definition in the *Apple Event Registry: Standard Suites.* Among these characteristics are properties, element classes, and Apple events that can specify objects of that class. An object class is specified in an object specifier record by a keyword-specified descriptor record with the keyword `keyAEDesiredClass` whose data handle refers to either a constant or an object class ID.

***object class ID*** A four-character code, which can also be represented by a constant, that identifies an object class for an Apple event object. The object class ID for a primitive object class is the same as the four-character value of its descriptor type.

***object class inheritance hierarchy*** The hierarchy of subclasses and superclasses that determines which properties, elements, and Apple events object classes inherit from other object classes.

***object-comparison function*** An object callback function that compares an element to either another element or to a descriptor record and returns either `TRUE` or `FALSE`.

***object-counting function*** An object callback function that counts the number of elements of a specified class in a specified container, so that the Apple Event Manager can determine how many elements it must examine to find the element or elements that pass a test.

***object-marking function*** An object callback function called repeatedly by the Apple Event Manager to mark specific Apple event objects. See also ***marking callback functions.***

***object specifier record*** A description of one or more Apple event objects based on the Apple Event Manager and the classification system defined in the *Apple Event Registry: Standard Suites.* An object specifier record consists of a descriptor record of descriptor type `typeObjectSpecifier` that comprises four keyword-specified descriptor records: the object class ID, the container for the Apple event object (which is usually another Apple event object, specified by another object specifier record), the key form, and the key data.

***Open Application event*** An Apple event that asks an application to perform the tasks—such as displaying untitled windows—associated with opening itself; one of the four required Apple events.

***Open Documents event*** An Apple event that asks an application to open one or more documents specified in a list; one of the four required Apple events.

***Open Scripting Architecture (OSA)*** A mechanism based on the Apple Event Manager and the *Apple Event Registry: Standard Suites* that allows users to control multiple applications by means of scripts. The scripts can be written in any scripting language that supports the OSA.

***optional parameter*** A supplemental parameter in an Apple event used to specify data that the server application can use in addition to the data specified in the direct parameter. Source applications list the keywords for parameters that they consider optional in the attribute identified by the `keyOptionalKeywordAttr` keyword. Target applications use this attribute to identify any parameters that they are required to understand. If a parameter's keyword is not listed in this attribute, the target application must understand that parameter to handle the event successfully. See also ***Apple event attribute, Apple event parameter.***

***port*** (1) A portal through which an open application can exchange information with another open application using the PPC Toolbox. A port is designated by a port name and a location name. An application can open as many ports as it requires so long as each port name is unique within a particular computer. (2) A connection between the CPU and main memory or a device (such as a terminal) for transferring data. (3) A socket on the back panel of a computer where you plug in a cable for connection to a network or a peripheral device.

***port name*** A unique identifier for a particular application within a computer. The port name contains a name string, a type string, and a script code. An application can specify any number of port names for a single port so long as each name is unique. See also ***port.***

***primitive object class*** An object class defined in the *Apple Event Registry: Standard Suites* for Apple event objects that contain a single value; for example, the `cBoolean`, `cLongInteger`, and `cAlias` object classes are all primitive object classes. An Apple event object that belongs to a primitive object class has no properties and contains only one element—the value of the data.

***Print Documents event*** An Apple event that requests that an application print a list of documents; one of the four required Apple events.

***Program-to-Program Communications (PPC) Toolbox*** The collection of routines that allows applications to exchange blocks of data with other applications by reading and writing low-level message blocks.

***property*** An Apple event object that defines some characteristic of another Apple event object, such as its font or point size, that can be uniquely identified by a constant. The definition of each object class in the *Apple Event Registry: Standard Suites* lists the constants and class IDs for properties of Apple event objects belonging to that object class. For example, the constants `pName` and `pBounds` identify the name and boundary properties of Apple event objects that belong to the object class `cWindow`. The `pName` property of a specific window is defined by an Apple event object of object class `cProperty`, such as the word "MyWindow," which defines the name of the window. An Apple event object can contain only one of each of its properties, whereas it can contain many elements of the same element class. See also ***Apple event object, container, element classes.***

***property ID*** A four-character code, which can also be represented by a constant, that identifies a property.

***publish*** To make data available to other documents and applications through a publisher. When a user creates or edits the data in the publisher and then saves it, the current version of the data is stored in an edition. See also ***edition, publisher, subscriber.***

***publisher*** A portion of a document that makes its data available to other documents or applications. A publisher stores its data in an edition whenever a user creates or edits the data in the publisher and then saves it. See also ***edition, section, subscriber.***

***query*** A string of commands and data sent to a database or other data source. A query does not necessarily extract data from a data source; it might only send data or commands to a database or other application.

***query definition function*** A function contained in a ***query document*** that prompts the user for information and modifies the query before the Data Access Manager sends it to the data server.

***query document*** A file of file type `'qery'` containing commands and data in a format appropriate for a database or other data source. An application uses high-level Data Access Manager routines to open a query document.

***query record*** A data structure in memory containing information provided by a `'qrsc'` resource. The query record includes a pointer to a query.

***Quit Application event*** An Apple event that requests that an application perform the tasks—such as releasing memory, asking the user to save documents, and so on—associated with quitting; one of the four required Apple events. The Finder sends this event to an application immediately after sending it a Print Documents event or if the user chooses Restart or Shut Down from the Finder's Special menu.

***range descriptor record*** A coerced AE record of type `typeRangeDescriptor` that identifies two Apple event objects marking the beginning and end of a range of elements. See also ***boundary objects.***

***recordable application*** An application that uses Apple events to report user actions to the Apple Event Manager for recording purposes. When a user turns on recording (for example, by pressing the Record button in the Script Editor application), a scripting component translates the Apple events generated by the user's subsequent actions into statements in a scripting language and records them in a compiled script. See also ***scriptable application.***

***recordable event*** Any Apple event that any recordable application sends to itself while recording is turned on for the local computer, with the exception of events that are sent with the `kAEDontRecord` flag set in the `sendMode` parameter of the `AESend` function.

***recording process*** Any process (for example, a script editor) that can turn Apple event recording on and off and receive and record recordable Apple events.

***required Apple event*** One of the four Apple events in the Required suite that the Finder sends to applications: Open Documents, Open Application, Print Documents, or Quit Application.

***required parameter*** An Apple event parameter that must be included in an Apple event. For example, a list of documents to open is a required parameter for the Open Documents event. Direct parameters are often required, and other additional parameters may be required. Optional parameters are never required.

***resolve*** To locate the Apple event object described by an object specifier record.

***result handler*** A routine that the Data Access Manager calls to convert a data item to a character string.

***results record*** A structure that the Data Access Manager uses to store the data retrieved by the `DBGetQueryResults` function. This data is returned by a data source in response to a query.

***resume dispatch function*** An application-defined function called by `OSADoEvent` or `OSAExecuteEvent` to dispatch an Apple event directly to an application's default handler for that event.

***script*** Any collection of data that, when executed by the appropriate program, causes a corresponding action or series of actions. When a scripting component that supports the OSA executes a script, it sends Apple events as necessary to trigger actions in server applications.

***scriptable application*** An application that can respond as a server application to Apple events sent to it by scripting components. To be scriptable, an application must respond to the appropriate standard Apple events, and it must provide an `'aete'` resource that describes the nature of that support. See also ***Apple event user terminology resources.***

***script application*** A script file with the file type `'APPL'` that contains the script data as a resource of type `'scpt'`. If a script application has the creator signature `'aplt'`, a user can double-click its icon to trigger the script. If a script application has the creator signature `'dplt'`, a user can drag the icon for another file or a folder over the script application's icon to trigger its script. By default, when a user triggers the script in a script application, a splash screen appears that allows the user either to quit or to run the script. Users can also save a script application in a form that bypasses the splash screen, running the script immediately after the user double-clicks its icon.

***script application component*** A component registered with the Component Manager at system startup. When a user opens a script application, the script application component loads the script and passes the resulting script ID to the appropriate scripting component for execution.

***script comment*** A description, in a script editor window, of what the script displayed in that window does.

***script context*** A form of script that maintains context information for the execution of other scripts. A script context can also be used to handle Apple events. Like a compiled script, a script context can be decompiled as source data. In the AppleScript scripting language, a script context is called a `script object`.

***script data*** A compiled script, script value, script context, or any other representation of a script in memory used internally by a scripting component. See also ***compiled script, script context, script value.***

***script editor*** An application that allows users to record, edit, save, and execute scripts; for example, the Script Editor application provided with AppleScript.

***script file*** A file in which a script is stored. A script file can be a compiled script file, a script application file, or a script text file.

***script ID*** A data structure of type `OSAID`—that is, a long integer—used by scripting components to keep track of script data.

***scripting*** Writing and executing scripts to control the behavior of multiple applications.

***scripting component*** A component that responds appropriately to calls made to the standard scripting component routines. Most scripting components implement scripting languages; for example, the AppleScript component implements the AppleScript scripting language.

***script object*** AppleScript term for script context. See also ***script context.***

***script text file*** Uncompiled statements in a scripting language saved by a script editor as a text file. A user must open a script text file in a script editor and successfully compile it before it will execute. See also ***script editor.***

***script value*** An integer, a string, a Boolean value, a constant, a `'PICT'`, or any other fixed data that a scripting component returns or uses in the course of executing a script.

***section*** A document or portion of a document that shares its contents with other documents. The Edition Manager supports two types of sections: publishers and subscribers. A publisher makes its data available to share and a subscriber subscribes to available data. See also ***publisher, subscriber.***

***send function*** A function called by a scripting component whenever it sends an Apple event during script execution. You can provide an alternative send function if you want your application to perform some action instead of or in addition to sending Apple events. If you don't provide an alternative send function, scripting components call the standard Apple Event Manager function `AESend` with default parameters.

***server application*** An application that responds to Apple events requesting a service or information sent by client applications or scripting components (for example, by printing a list of files, checking the spelling of a list of words, or performing a numeric calculation). Apple event servers and clients can reside on the same local computer or on remote computers connected to a network.

***session*** (1) A logical (as opposed to physical) connection between two entities (such as a Macintosh program and a database server) that facilitates the transmission of information between the two entities. (2) In the PPC Toolbox, an exchange of information between one open application with a port and another open application with a port. Sessions can occur between applications that are located on the same computer or across a network. An application has the option to accept or reject a session request. Authentication of the requesting user may be required before a session can commence. See also ***authentication, message block, port.***

***session ID*** A number that uniquely identifies a ***session.***

***source application*** The application that sends a particular Apple event to another application or to itself. Typically, an Apple event client sends an Apple event requesting a service from an Apple event server; in this case, the client is the source application for the Apple event. The Apple event server may return a different Apple event as a reply; in this case, the server is the source for the reply Apple event.

***source data*** Statements in a scripting language that constitute an uncompiled script.

***special handler dispatch table*** A table in either the application heap or the system heap that the Apple Event Manager uses to keep track of various specialized handlers.

***status routine*** An application-defined routine that can update windows, check the results of the low-level calls made by the Data Access Manager's `DBStartQuery` and `DBGetQueryResults` functions, and cancel execution of these functions when appropriate to do so.

***subclass*** An object class that inherits properties, element classes, and Apple events from another object class—its superclass. A subclass can also include properties, element classes, or Apple events that are not inherited from its superclass. Every object class, with the exception of `cObject`, is a subclass of another object class. See also ***object class, superclass.***

***subscribe*** To obtain data that a publisher makes available in an edition. A user subscribes to a publisher by choosing Subscribe To from the Edit menu and selecting the desired edition. See also ***edition, publish.***

***subscriber*** A portion of a document that automatically obtains current data from other documents and applications. A subscriber reads data from an edition. See also ***edition, section.***

***suite*** In the *Apple Event Registry: Standard Suites,* a group of definitions for Apple events, object classes, primitive object classes, descriptor types, and constants that are all used for a set of related activities. For example, the Text suite includes definitions of Apple events, object classes, and so on that are used for text processing.

***superclass*** The object class from which a subclass inherits properties, elements, and Apple events. See also ***object class, subclass.***

***system Apple event dispatch table*** See ***Apple event dispatch table.***

***system coercion dispatch table*** See ***coercion handler dispatch table.***

***system object accessor dispatch table*** See ***object accessor dispatch table.***

***system result handler*** A result handler that is available to all applications that use the system. Compare with ***application result handler.***

***target address*** An application signature, a process serial number, a session ID, a target ID record, or some other application-defined type that identifies the target of an Apple event.

***target application*** The application addressed to receive an Apple event. Typically, an Apple event client sends an Apple event requesting a service from a server application; in this case, the server is the target application of the Apple event. The server application may return a different Apple event as a reply; in this case, the client is the target of the reply Apple event.

***token*** A descriptor record returned by an object accessor function that identifies a requested Apple event object in a specified container.

***token disposal function*** An object callback function that disposes of a token.

***transaction*** A sequence of Apple events sent back and forth between the client and server applications, beginning with the client's initial request for a service. All Apple events that are part of one transaction must have the same transaction ID.

***whose descriptor record*** A coerced AE record of descriptor type `typeWhoseDescriptor`. The Apple Event Manager creates whose descriptor records when it resolves object specifier records that specify `formTest`.

***whose range descriptor record*** A coerced AE record of type `typeWhoseRange`. Under certain conditions, the Apple Event Manager coerces a range descriptor record to a whose range descriptor record when it resolves object specifier records that specify `formTest`.