# Figures, Tables, and Listings

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Preface</strong></td>
<td></td>
<td>xxiii</td>
</tr>
<tr>
<td><strong>Chapter 1</strong></td>
<td>Introduction to Interapplication Communication</td>
<td>1-1</td>
</tr>
<tr>
<td>Figure 1-1</td>
<td>Principal methods of communication between applications</td>
<td>1-5</td>
</tr>
<tr>
<td>Figure 1-2</td>
<td>Sharing data with the aid of the Edition Manager</td>
<td>1-7</td>
</tr>
<tr>
<td>Figure 1-3</td>
<td>A publisher, an edition, and a subscriber</td>
<td>1-8</td>
</tr>
<tr>
<td>Figure 1-4</td>
<td>Sharing dynamic data with other applications</td>
<td>1-8</td>
</tr>
<tr>
<td>Figure 1-5</td>
<td>Sending and responding to Apple events with the aid of the Apple Event Manager</td>
<td>1-10</td>
</tr>
<tr>
<td>Figure 1-6</td>
<td>A Set Data event</td>
<td>1-12</td>
</tr>
<tr>
<td>Figure 1-7</td>
<td>How a scripting component executes a script</td>
<td>1-14</td>
</tr>
<tr>
<td>Figure 1-8</td>
<td>A Set Data event sent during script execution</td>
<td>1-17</td>
</tr>
<tr>
<td>Figure 1-9</td>
<td>Recording user actions in a factored application</td>
<td>1-19</td>
</tr>
<tr>
<td>Figure 1-10</td>
<td>Controlling an application's own behavior by executing a script</td>
<td>1-20</td>
</tr>
<tr>
<td>Figure 1-11</td>
<td>Posting an invoice and updating a database by executing a script</td>
<td>1-21</td>
</tr>
<tr>
<td><strong>Chapter 2</strong></td>
<td>Edition Manager</td>
<td>2-1</td>
</tr>
<tr>
<td>Figure 2-1</td>
<td>The default edition icon</td>
<td>2-4</td>
</tr>
<tr>
<td>Figure 2-2</td>
<td>A publisher, an edition, and a subscriber</td>
<td>2-5</td>
</tr>
<tr>
<td>Figure 2-3</td>
<td>The publisher dialog box</td>
<td>2-5</td>
</tr>
<tr>
<td>Figure 2-4</td>
<td>The subscriber dialog box</td>
<td>2-7</td>
</tr>
<tr>
<td>Figure 2-5</td>
<td>A document and its corresponding editions</td>
<td>2-8</td>
</tr>
<tr>
<td>Figure 2-6</td>
<td>Publisher and subscriber borders</td>
<td>2-9</td>
</tr>
<tr>
<td>Figure 2-7</td>
<td>Edition Manager commands in the Edit menu</td>
<td>2-10</td>
</tr>
<tr>
<td>Figure 2-8</td>
<td>Edition Manager commands under the Publishing menu command</td>
<td>2-11</td>
</tr>
<tr>
<td>Listing 2-1</td>
<td>Accepting Section Read events and verifying if a section is registered</td>
<td>2-14</td>
</tr>
<tr>
<td>Figure 2-9</td>
<td>A document with a publisher and subscriber and its resource fork</td>
<td>2-16</td>
</tr>
<tr>
<td>Figure 2-10</td>
<td>The new publisher alert box</td>
<td>2-19</td>
</tr>
<tr>
<td>Listing 2-2</td>
<td>Saving a document containing sections</td>
<td>2-21</td>
</tr>
<tr>
<td>Listing 2-3</td>
<td>Opening a document containing sections</td>
<td>2-23</td>
</tr>
<tr>
<td>Figure 2-11</td>
<td>A sample publisher dialog box</td>
<td>2-29</td>
</tr>
<tr>
<td>Listing 2-4</td>
<td>Creating a publisher</td>
<td>2-33</td>
</tr>
<tr>
<td>Listing 2-5</td>
<td>Writing data to an edition</td>
<td>2-36</td>
</tr>
<tr>
<td>Figure 2-12</td>
<td>A sample subscriber dialog box</td>
<td>2-37</td>
</tr>
<tr>
<td>Listing 2-6</td>
<td>Creating a subscriber</td>
<td>2-40</td>
</tr>
<tr>
<td>Listing 2-7</td>
<td>Reading in edition data</td>
<td>2-42</td>
</tr>
</tbody>
</table>
Chapter 3

Introduction to Apple Events  3-1

Figure 3-1  An Open Documents event  3-4
Figure 3-2  A Get Data event  3-7
Figure 3-3  Major attributes and direct parameter of an Open Documents event  3-10
Figure 3-4  Major attributes and direct parameter of a Get Data event  3-11
Figure 3-5  A descriptor record whose data handle refers to an unterminated string  3-13
Figure 3-6  A descriptor record whose data handle refers to event class data  3-14
Figure 3-7  A keyword-specified descriptor record for the event class attribute of an Open Documents event  3-16
Figure 3-8  A descriptor list for a list of aliases  3-17
Figure 3-9  Data structures within an Open Documents event  3-19
Figure 3-10  Accepting and processing an Open Documents event  3-21
Figure 3-11  The Apple Event Manager calling the handler for an Open Documents event  3-22
Figure 3-12  Responding to an Open Documents event  3-27
Figure 3-13  Data structures within a simple object specifier record  3-37
Figure 3-14  An object specifier record in a Get Data event  3-38
Figure 3-15  Superclasses and subclasses  3-40
Figure 3-16  The object class inheritance hierarchy for the object class cWindow  3-44
Figure 3-17  An Apple event object of class cWord contained in an Apple event object of class cParagraph  3-46

Listing 2-8  Responding to action codes  2-46
Figure 2-17  Edit menu with Show/Hide Borders menu command  2-51
Figure 2-18  Publisher borders  2-52
Figure 2-19  Subscriber borders  2-53
Figure 2-20  A publisher with contents removed  2-54
Figure 2-21  A publisher border within a spreadsheet document  2-55
Figure 2-22  A publisher border with resize handles  2-56
Figure 2-23  A publisher and subscriber with clipped graphics  2-57
Figure 2-24  Creating multiple publishers alert box  2-58
Figure 2-25  Saving multiple publishers alert box  2-58
Figure 2-26  Subscribing directly to a 'PICT' file  2-62
Listing 2-9  Using your own edition opener function  2-67
### Chapter 4: Responding to Apple Events

| Listing 4-1 | A `DoEvent` procedure | 4-5 |
| Listing 4-2 | A `DoHighLevelEvent` procedure for handling Apple events and other high-level events | 4-6 |
| Listing 4-3 | Adding entries for the required Apple events to an application's Apple event dispatch table | 4-8 |
| Listing 4-4 | Adding entries for Apple events sent by the Edition Manager to an application's Apple event dispatch table | 4-9 |
| Listing 4-5 | A handler for the Open Application event | 4-15 |
| Listing 4-6 | A handler for the Open Documents event | 4-15 |
| Listing 4-7 | A handler for the Print Documents event | 4-18 |
| Listing 4-8 | A handler for the Quit Application event | 4-19 |
| Listing 4-9 | A handler for the Create Publisher event | 4-23 |
| Listing 4-10 | Extracting items from a descriptor list | 4-33 |
| Listing 4-11 | A function that checks for a `keyMissedKeywordAttr` attribute | 4-35 |
| Listing 4-12 | Adding the `keyErrorString` parameter to the reply Apple event | 4-38 |
| Listing 4-13 | Adding parameters to the reply Apple event | 4-39 |
| Table 4-1 | Coercion handling provided by the Apple Event Manager | 4-43 |
| Listing 4-14 | Using the `AEInteractWithUser` function | 4-50 |
| Figure 4-1 | A document with a button that triggers a Get Data event | 4-51 |
| Figure 4-2 | A server application displaying a dialog box that requests information from the user | 4-52 |
| Figure 4-3 | Handling user interaction | 4-53 |
| Figure 4-4 | Handling user interaction with the `kAEWaitReply` flag set | 4-54 |
| Figure 4-5 | Handling user interaction with the `kAEQueueReply` flag set | 4-55 |
| Table 4-2 | Descriptor types used by the Apple Event Manager (excluding those used with object specifier records) | 4-57 |

### Chapter 5: Creating and Sending Apple Events

| Listing 5-1 | Creating the optional keyword for the Create Publisher event | 5-9 |
| Listing 5-2 | Creating a target address | 5-11 |
| Listing 5-3 | Specifying a target address in an Apple event by using the `PPCBrowser` function | 5-12 |
| Listing 5-4 | Sending an Apple event | 5-18 |
| Listing 5-5 | An idle function | 5-23 |

### Chapter 6: Resolving and Creating Object Specifier Records

| Figure 6-1 | Resolving an object specifier record for a table in a document | 6-6 |
| Figure 6-2 | Nested object specifier records that specify a container hierarchy | 6-10 |
| Table 6-1 | Standard descriptor types used with `keyAEKeyData` | 6-12 |
| Table 6-2 | Keyword-specified descriptor records for `typeCompDescriptor` | 6-16 |
Table 6-3  Keyword-specified descriptor records for
    typeLogicalDescriptor  6-17
Figure 6-3  The container hierarchy for the first row in a table that meets a
test  6-18
Figure 6-4  A logical descriptor record that specifies a test  6-19
Table 6-4  Keyword-specified descriptor records in a descriptor record of type
    typeRangeDescriptor  6-20
Listing 6-1  Installing object accessor functions that find elements of different
classes for container tokens of the same type  6-23
Listing 6-2  Installing one object accessor function that finds elements of
different classes for container tokens of one type  6-25
Listing 6-3  Installing object accessor functions that find elements of the same
class for container tokens of different types  6-25
Listing 6-4  Installing object accessor functions that locate elements of different
classes in the default container  6-26
Listing 6-5  An object accessor function that locates Apple event objects of
    object class cDocument  6-30
Listing 6-6  An object accessor function that locates Apple event objects of
    object class cParagraph  6-32
Listing 6-7  An object accessor function that locates Apple event objects of
    object class cWord  6-34
Listing 6-8  An object accessor function that locates Apple event objects of
    object class cWindow  6-35
Listing 6-9  An object accessor function that identifies any property of a
    window  6-38
Figure 6-5  Descriptor record for an application-defined token that identifies a
document  6-39
Figure 6-6  Descriptor record for an application-defined token that identifies the
    pBounds property of a window  6-40
Table 6-5  Keyword-specified descriptor records for
    typeWhoseDescriptor  6-42
Figure 6-7  A container hierarchy created by the Apple Event Manager using a
    whose descriptor record  6-43
Table 6-6  Keyword-specified descriptor records for
    typeWhoseRange  6-44
Listing 6-10  An object-counting function  6-49
Listing 6-11  Object-comparison function that compares two Apple event
    objects  6-52
Table 6-7  Nested object specifier records that describe a container
    hierarchy  6-56
Listing 6-12  Creating an object specifier record using
    CreateObjSpecifier  6-58
Listing 6-13  Using CreateObjSpecifier in an application-defined function
    6-59
Listing 6-14  Specifying a document container  6-61
Listing 6-15  Specifying a table container  6-62
Table 6-8  Object specifier record for the first row that meets a test in the table
    named “MyAddresses”  6-65
Table 6-9  Logical descriptor record that specifies a test  6-66
Listing 6-16  Creating an object specifier record with the key form
    firstName  6-67
Listing 6-17  Creating a comparison descriptor record  6-68
Listing 6-18  Creating a logical descriptor record  6-70
Listing 6-19  Creating a complex object specifier record  6-70
Table 6-10  A range descriptor record 6-73
Listing 6-20  Creating a range descriptor record 6-74
Table 6-11  Keyword-specified descriptor records for typeObjectSpecifier 6-76

Chapter 7  Introduction to Scripting 7-1
Figure 7-1  A script window in the Script Editor application 7-6
Figure 7-2  Script file icons in the Finder and corresponding user actions 7-7
Figure 7-3  How the AppleScript component executes a script 7-9
Figure 7-4  How an application uses the AppleScript component to execute a script 7-13
Figure 7-5  Role of the 'aete' and 'aeut' resources when the AppleScript component compiles and executes a script 7-18
Figure 7-6  Role of the 'aete' and 'aeut' resources when the AppleScript component records and decompiles a script 7-19
Figure 7-7  Using a handler in a script context to handle an Apple event 7-26

Chapter 8  Apple Event Terminology Resources 8-1
Table 8-1  Syntax for AppleScript arguments that correspond to direct parameters 8-5
Table 8-2  Syntax for AppleScript arguments that correspond to insertion location descriptor records 8-6
Table 8-3  Structure of the 'aete' and 'aete' resources 8-8
Listing 8-1  Resource type declaration for the 'aete' resource 8-9
Listing 8-2  Rez input for an 'aete' resource for an application that supports the Required and Core suites in their entirety 8-15
Listing 8-3  Rez input for an 'aete' resource that extends the definitions of the Required, Core, and Text suites 8-17
Listing 8-4  A handler for the Get AETE event 8-25
Figure 8-1  Structure of an 'aete' or 'aete' resource 8-26
Figure 8-2  Structure of the header data in an 'aete' or 'aete' resource 8-27
Figure 8-3  Structure of suite data in an 'aete' or 'aete' resource 8-28
Figure 8-4  Structure of event data in an 'aete' or 'aete' resource 8-30
Figure 8-5  Structure of additional parameter data in an 'aete' or 'aete' resource 8-34
Figure 8-6  Structure of object class data in an 'aete' or 'aete' resource 8-36
Figure 8-7  Structure of property data in an 'aete' or 'aete' resource 8-38
Figure 8-8  Structure of element class data in an 'aete' or 'aete' resource 8-41
Figure 8-9  Structure of comparison operator data in an 'aete' or 'aete' resource 8-42
Figure 8-10 Structure of enumeration data in an 'aeut' or 'aete' resource  8-43
Figure 8-11 Structure of enumerator data in an 'aeut' or 'aete' resource  8-44
Listing 8-5 Resource type declaration for the 'scsz' resource  8-45

Chapter 9  Recording Apple Events  9-1
Listing 9-1 A function used by a factored application to send itself a Quit Application event  9-7
Listing 9-2 A routine used by a factored application to handle a Quit Application event  9-8
Listing 9-3 A routine used by a factored application to send itself a Create Element event  9-10
Listing 9-4 The Create Element event handler for a factored application  9-11
Listing 9-5 A routine used by a factored application to handle window movement  9-13

Chapter 10  Scripting Components  10-1
Listing 10-1 Locating a scripting component that supports specific optional routines  10-6
Listing 10-2 A routine that compiles and executes source data  10-9
Listing 10-3 A procedure that uses OSAScriptError to get information about an execution error  10-11
Figure 10-1 A generic storage descriptor record  10-12
Figure 10-2 A component-specific storage descriptor record  10-13
Listing 10-4 A routine that loads and executes script data previously saved using a generic storage descriptor record  10-16
Listing 10-5 A routine that displays a compiled script for editing and recompiles it  10-18
Listing 10-6 A function that loads and modifies script data, then saves it using a generic storage descriptor record  10-19
Listing 10-7 A general Apple event handler that uses the OSADoEvent function  10-21

Chapter 11  Program-to-Program Communications Toolbox  11-1
Figure 11-1 A PPC Toolbox session between two applications  11-5
Figure 11-2 The icon for the Sharing Setup control panel  11-6
Figure 11-3 The Sharing Setup control panel  11-6
Figure 11-4 The session termination alert box  11-7
Figure 11-5 The users and groups dialog box  11-8
Figure 11-6 The user termination alert box  11-8
Figure 11-7 The guest dialog box  11-9
Figure 11-8 The PPC Toolbox authentication process  11-10
Listing 11-1 Initializing the PPC Toolbox using the PPCInit function  11-12
Figure 11-9 Database and spreadsheet applications using the PPC Toolbox  11-13
Figure 11-10 Two Macintosh applications and their corresponding ports 11-18
Figure 11-11 The PPC Toolbox and a dictionary service application 11-20
Listing 11-2 Opening a PPC port 11-21
Figure 11-12 The program linking dialog box 11-22
Figure 11-13 The program linking dialog box without a zone list 11-23
Listing 11-3 Using a port filter function 11-24
Listing 11-4 Browsing through dictionary service ports 11-26
Listing 11-5 Using the IPCListPorts function to obtain a list of ports 11-28
Figure 11-14 The user identity dialog box 11-30
Figure 11-15 The incorrect password dialog box 11-31
Figure 11-16 The invalid user name dialog box 11-31
Listing 11-6 Using the StartSecureSession function to establish a session 11-32
Listing 11-7 Initiating a session using the PPCStart function 11-34
Listing 11-8 Using the PPCInform function to enable a port to receive sessions 11-36
Listing 11-9 Completion routine for a PPCInform function 11-37
Listing 11-10 Accepting a session request using the PPCAccept function 11-38
Listing 11-11 Completion routine for a PPCAccept function 11-38
Listing 11-12 Rejecting a session request using the PPCReject function 11-39
Listing 11-13 Completion routine for a PPCReject function 11-39
Figure 11-17 Transmitting message blocks 11-40
Listing 11-14 Using the PPCRead function to read data during a session 11-41
Listing 11-15 Polling the ioResult field to determine if a PPCRead function has completed 11-41
Listing 11-16 Using the PPCWrite function to write data during a session 11-42
Listing 11-17 Polling the ioResult field to determine if a PPCWrite function has completed 11-43
Listing 11-18 Ending a PPC session using the PPCEnd function 11-43
Listing 11-19 Closing a PPC port using the PPCClose function 11-44
Listing 11-20 Using the DeleteUserIdentity function to invalidate a user identity 11-45
Figure 11-18 The PPC Toolbox parameter blocks 11-47

Chapter 12 Data Access Manager 12-1

Figure 12-1 A connection with a database 12-6
Figure 12-2 Using high-level Data Access Manager routines 12-8
Figure 12-3 Using low-level Data Access Manager routines 12-10
Figure 12-4 A flowchart of a session using the high-level interface 12-15
Listing 12-1 Using the high-level interface 12-18
Listing 12-2 Two completion routines 12-21
Listing 12-3 A sample status routine 12-26
Figure 12-5 A flowchart of a session using the low-level interface 12-30
Listing 12-4 Sending a query fragment 12-32
Listing 12-5 Using the low-level interface 12-34
Table 12-1  Data types defined by the Data Access Manager  12-39
Listing 12-6  A result handler  12-46
Figure 12-6  A query document dialog box  12-48
Figure 12-7  The relationship between resources in a query document and the query record  12-50
Figure 12-8  The relationship between a query definition function and queries  12-51
Listing 12-7  A query definition function  12-53
Figure 12-9  Structure of a compiled query (‘qrsc’) resource  12-91
Figure 12-10  Structure of a compiled query string (‘wstr’) resource  12-92