iWork Programming Guide

2005-11-09

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

Figures, Tables, and Listings

Introduction to iWork Programming Guide

Welcome to the suite of iApps productivity tools called iWork.

Keynote and Pages form the foundation of the iWork suite. Both are designed to provide Apple users with easy to use, yet powerful, feature-rich graphics applications running in Mac OS X. The latest version of Keynote, for example, lets users create high-quality, professional-looking presentations, offering a wide variety of themes, chart and table templates, as well as animated slide transitions. Pages offers users a wide-range of custom templates for easy page layout and design.

Developers can take advantage of Apple's XML format for Keynote files, known as APXL, to directly create or modify their Keynote presentations. Likewise, developers can create custom templates or modify existing Pages documents by accessing the Pages file format.

Each application is defined by an XML schema. Because both Keynote and Pages use an open, easily accessible file format, developers have a unique opportunity to enhance their existing products or even create new ones, using the iWork XML schemas.

These schemas provide access to the contents of Keynote 2.x and Pages 1.x documents, including themes and slide lists, page layouts, and other organizational structures. Using either of these formats, you can modify the content of your document in ways that supplement the capabilities of either or both applications.

Who Should Read This Document

This document explains, at a basic level, how each XML schema works and offers some examples of common usage. You might find this information valuable if you need to create, modify, or query Keynote or Pages XML documents. If you simply need to parse XML and extract information from an existing source of XML, you may need to look elsewhere for examples that are more suited to your needs.

Important: The information in this programming guide is intended to supersede the following Apple technical documents:

Technical Note TN2067, available at http://developer.apple.com/technotes/tn2002/tn2067.html, which lists the XML schema file that defines the syntax of a Keynote 1.x APXL file. With the introduction of Keynote 2.x, this schema file is out of date.

Technical Note TN2073, available at http://developer.apple.com/technotes/tn2002/tn2073.html, which describes in detail how to create a slide-list in a sample Keynote 1.x presentation. Although the information in this technical note may appear useful, you should not rely on it for developing or modifying your own products.

Both technical notes are no longer applicable for developers who want to work with Keynote 2.x presentations or with Pages 1.x layouts and designs.

This document does *not* describe the complete XML schema for either Pages 1.x or Keynote 2.x. The complete XML schema for both applications is not available and will not be made public. Nor is this document, by any stretch of the imagination, intended as a comprehensive tutorial on how to customize or extend third-party applications that rely on the schemas of each iWork application. The focus here is on bringing developers up to date on how the schemas work, and what they may need to know in order to take advantage of the robust feature sets available in each application.

Important: This document only covers the file formats for Keynote 2.x and Pages 1.x. Future versions of those products may use a different file format than the ones described here. Developers should understand that Apple cannot guarantee that the file formats described herein will be supported in those future versions of the iWork applications as they are currently supported. Changes to these file formats ought to be expected.

To work with this document, you should be familiar with standard XML conventions and be familiar with either one or both iWork applications.

Organization of This Document

This document begins with a high-level discussion of the iWork schemas and is followed by successive chapters that describe the various features of each iWork application, with deconstructed examples of how the XML is formed. Each chapter depends, to a certain extent, on understanding the material presented in any previous chapters.

- "The iWork XML Schemas" (page 11) describes the document structure of the iWork XML schemas used by Keynote 2.x and Pages 1.x.
- "Deconstructing iWork Shapes and Drawable Objects" (page 25) discusses the XML used in various iWork shapes, specifically shapes used in Keynote. The basic structure of iWork drawable objects is also defined. All of the shapes in the iWork XML shemas share certain common structures, such as geometry, style, binary, path, image, and so on.
- "Working With Text Objects in Pages Documents" (page 43) discusses how Pages text objects are stored and laid out in XML.

See Also

There are a wide range of resources available for developers working with XML. Here is a brief list of some of more useful ones:

- *Beginning XML*, Second Edition (various authors; published by Wrox Press)
- *XML in a Nutshell*, Second Edition (by Elliotte Rusty Harold and W. Scott Means; published by O'Reilly)
- http://www.w3schools.com/xml/default.asp offers concise tutorials on XML and related technologies
- The Apache XML Project: Apache is the home of Xerxes (XML parser for C++ and Java) and Xalan (XSLT processor for C++ and Java), both of which can be included in commercial software

The following are also useful websites on XML:

- The Annotated XML Specification—http://www.xml.com/axml/testaxml.htm
- O'Reilly XML.com—http://www.xml.com/

The iWork XML Schemas

Extensible Markup Language (XML) is a ubiquitous and flexible markup standard for processing and exchanging data. You can find XML in a wide range of categories, including property lists and file formats for various applications. XML is used extensively to specify the format of various sources of information on the Internet, including web-based services. XML is at the heart of both iWork applications, Keynote and Pages, developed by Apple.

This chapter provides you with a high-level, architectural view of the document-based XML schemas used by both Keynote 2.x and Pages 1.x. As noted in the "Introduction" (page 7), Apple's technical documents describing the Keynote XML schema pertain only to Keynote 1.x and are not relevant to any discussion of the current version of that application.

Important: This document does *not* provide you with a complete description of the file format specification or the XML schemas for Keynote 2.x and Pages 1.x, but rather with enough information to help in creating or modifying your application.

Developer Issues

Both Pages 1.x and Keynote 2.x, which form the basis of Apple's iWork package, are powerful, yet easy to use applications that have attracted large and ever-growing customer bases. The iWork package is part of a family of iApps productivity tools.

Because Keynote uses an open, easily accessible file format, developers can take advantage of its XML schema to enhance existing products or create new ones. Beginning with Keynote 1.x, it is possible to build applications that will create or change the contents of a Keynote presentation. For example, developers can use any of Keynote's themes and build an entire presentation of their own design simply by filling out the `<key:slide-list>` element in a text editor of their choice.

The Keynote APXL file is the engine that drives every presentation, specifying every detail of the presentation's appearance and behavior—from master slide and each individual slide to the transitions used between slides. The APXL file also defines the state of the presentation when the user first opens it. An understanding of how the elements in an APXL file combine to create a presentation is critical to developing applications that are robust and well-behaved.

Similarly, understanding the Pages document structure is important for Apple developers who want to customize their own page layouts or add value to their existing applications.

Important Changes to the File Format

When Apple introduced Keynote 2.x, a number of new features were added to the application. In so doing, the document file format was significantly expanded and revised.

For developers of existing applications that read and write the Keynote 1.x file format, this has necessitated updating their code to read and write the new Keynote 2.x format.

What this means is that Keynote 2.x will read existing Keynote 1.x files, so developers can continue creating documents in the older file format programmatically from their applications. *However, the original file will be converted to the new format and overwritten when it is saved from within Keynote 2.x.*

Important: While both Pages and Keynote XML schemas may lend themselves to developer modification and enhancements, a caveat is necessary here. Developers should not think of "extending the schemas" of either application, proceeding on the false or misleading assumption that they can add their own types of objects to the file formats of each, and consequently, have Keynote or Pages load those objects into their applications. This will simply not be the case, and is not recommended.

Note that the file type name of Keynote was changed from `presentation.apxl` in Keynote 1.x to `index.apxl` in Keynote 2.x.

Defining the iWork XML Schemas

A schema determines the layout of an XML document's elements, the attributes and subelements that each can have, and the constraints that the attribute data and element data must adhere to. XML schemas are used to define the structure, content and semantics of XML documents.

As with any XML file, Keynote and Pages XML consists of a series of elements, some of which may contain mixed content, while most include only attributes and/or child elements. Many elements have unique identifiers which allow them to be referenced by other elements.

Despite a number of internal differences, both iWork XML schemas represent an XML document as an ordered, labeled tree in which each node has a unique identity and may have a value, attributes, and namespaces associated with it.

The iWork XML schemas operate on the abstract, logical structure of an XML document—the data model—rather than its surface syntax. The data model represents an XML document as a tree of nodes. The tree can have various kinds of nodes, each of which corresponds to a type of XML construct, such as element, attribute, text, and namespace.

Each node in a tree typically has a unique identity. Most nodes have a name, and many have some string or other type of value associated with them.

To be usable in a particular context, an XML document must be *well formed*. This means that a document must have open and close tags for all its elements (in the correct sequence) and contain one root element. In addition, XML documents must have at least one XML declaration: an element that provides XML parsers with essential information needed to process a document and ensure that it is *valid*.

A valid document is one that follows the structure specified by a schema file, such as the Pages or Keynote schema files.

iWork XML Document Structures

This section discusses both the Keynote and Pages document structures, with a high-level architectural overview of the various elements and attributes that comprise each structure.

The Keynote Document Structure

If you deconstruct the XML for a valid Keynote presentation, you'll find that it conforms to an ordered hierarchical list. A Keynote XML document follows the basic hierarchy shown in "Listing 2-1," with the `<key:presentation>` element always at the top-level.

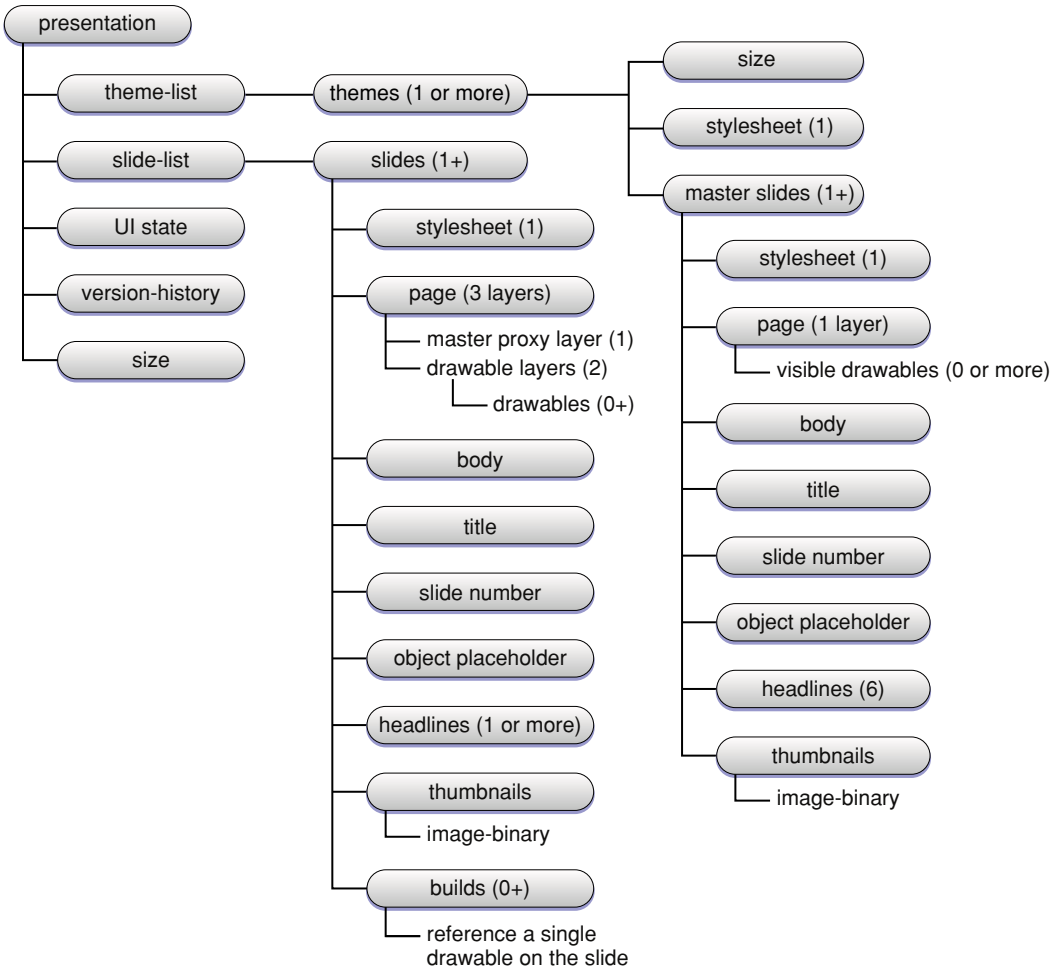
Listing 1-1 The Keynote XML hierarchy

```
<key:presentation>
    <key:theme-list>
        <key:theme>...</key:theme>
    </key:theme-list>
    <key:slide-list>
        <key:slides>...</key:slides>
    </key:slide-list>
    <key:ui-state>...</key:ui-state>
    <key:version-history>...</key:version-history>
    <key:size>...</key:size>
</key:presentation>
```

The Keynote XML format also specifies that the `<key:theme-list>` element must have at least one `<key:theme>` child element, and similarly, the `<key:slide-list>` element must have one or more `<key:slide>` child elements.

The hierarchy of the Keynote schema, with its ordering of elements and associated child elements, is illustrated in "Figure 2-1."

Figure 1-1 The Keynote hierarchy of elements



"Table 2-1" describes the usage of each Keynote child element.

Table 1-1 The Keynote presentation elements' children and their usage

Elements	Description
`<key:theme-list>`	Describes the default appearance of a Keynote document and each master slide. This element is made up of one or more themes, with child elements that include `<key:size>`, `<key:stylesheet>`, and `<key:master-sides>`.
`<key:slide-list>`	A list of slides, comprising `<key:stylesheet>`, `<key:page>`, `<key:body>`, `<key:title>`, `<key:slide-number>`, and other child elements. The `<key:slide-list>` element is at the center of any Keynote presentation: nothing else can appear in a `<key:slide-list>`. The order of slides in the list is the same as their order of appearance in a presentation. A Keynote document must contain at least one slide.

Elements	Description
<key:ui-state>	Describes the user interface state of a document; a required element, but can be left empty if you wish. The <key:ui-state> element allows you to set the following: which slide the user sees first when opening a document, the size of the document window, and the zoom level of the document.
<key:version-history>	Describes the elements that specify the version of the document.
<key:size>	Describes the dimensions of the presentation in pixels.

Defining Characteristics

In Keynote, the top-level <key:presentation> element is comprised of a <key:theme-list> and a <key:slide-list>. A <key:theme-list>, in turn, is comprised of several themes, including one <key:stylesheet> and one or more master-slides, and in turn, each master-slide has a single <key:stylesheet>.

Within this structure, there are zero or more visible drawables: <key:body>, <key:title>, <key:slide-number>, and <key:object-placeholder>, each of which may or may not be in a list of visible drawables.

A <key:slide-list> element is comprised of one or more slides. A <key:slide> consists of one <key:stylesheet> and three layers, one of which is a master proxy layer. A layer is a container for drawables. A master proxy layer displays all of the visible drawables from the slide's master-slide displayed in it. You can have one layer beneath or one layer over it, or just one layer over it. Every slide refers to one master-slide and each master-slide can have zero or more slides.

A <key:stylesheet> element for a slide inherits from the <key:stylesheet> for its master-slide. A <key:slide> element has three drawable layers.

Important: It is not recommended to have a <key:slide> element with either more or less than three drawable layers.

A slide also has a <key:body>, <key:title>, <key:slide-number>, and <key:object-placeholder>, which, just like the master slide, may or not be visible. If they are visible, they will show these layers. And a slide has one or more <key:headlines>, the first of which is the <key:title>, and is always present. But it will not appear unless the title placeholder is present.

A slide has a single page, and a page has three layers, one of which is the master proxy layer, two of which are drawable layers. A master slide has one page which has a single drawable layer. Master slides have exactly six headlines, which have no text, and are empty. Headlines each have depths. The six headlines on master slide always have depths 0, 1, 2, 3, 4, and 5. In addition, both master slides and slides have a <key:thumbnail> element that contains a single image binary.

A presentation has a <key:size> element that specifies the slide size. Each theme also has a <key:size> element.

Style and Stylesheet Inheritance

Style and style sheet inheritance are important because if you break the inheritance, your document will behave in undefined ways.

In Keynote, there are three levels of stylesheet: theme (which has no inheritance), master, and slide. Master stylesheets always inherit from a theme stylesheet. Slide stylesheets always inherit from a master stylesheet. For reasons of performance, you have to explicitly set up these inheritance relationships.

A slide style cannot inherit directly from the theme style. It has to inherit from another element in its stylesheet or its parents' master stylesheet.

A stylesheet's parent is indicated by the `<sf:parent-ref>` child element, where the value of the `sfa:IDREF` attribute is the value of the `sfa:ID` attribute of the parent stylesheet's `<key:stylesheet>` element.

The Pages Document Structure

The Pages document structure shares many substructures with the Keynote structure, with some notable exceptions, as discussed in this section. If you deconstruct the XML for a valid Pages document, you'll find that it conforms to an ordered hierarchical list shown in "Listing 2-2."

Listing 1-2 The Pages XML hierarchy

```
<sl:document>
    <sl:version-history>...</sl:version-history>
    <sl:publication-info>...</sl:publication-info>
    <sl:metadata>...</sl:metadata>
    <sl:print-info>...</sl:print-info>
    <sl:section-prototypes>
        <sl:prototype>
            <sl:stylesheet>...</sl:stylesheet>
            <sl:headers>...</sl: headers>
            <sl:footers>...</sl:footers>
            <sl:drawables>...</sl:drawables>
            <sl:text-storage>...</sl:text-storage>
            <sl:thumbnails>...</sl:thumbnails>
        </sl:prototype>
    <sl:/section-prototypes>
    <sl:stylesheet>...</sl:stylesheet>
    <sl:headers>...</sl: headers>
    <sl:footers>...</sl:footers>
    <sl:drawables>...</sl:drawables>
    <sl:text-storage>...</sl:text-storage>
    <sl:thumbnails>...</sl:thumbnails>
    <sl:window-configs>...</sl:window-configs>
</sl:document>
```

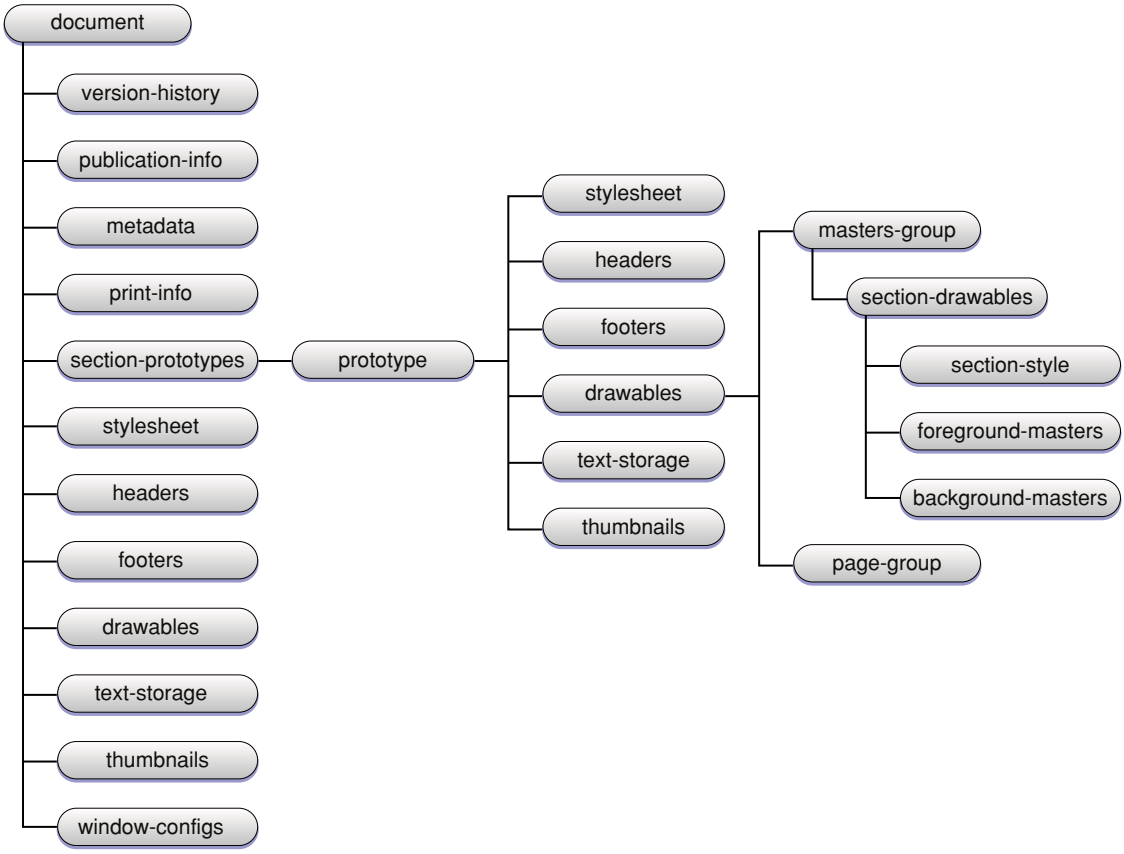
The Pages XML format specifies that the `<sl:document>` element is always at the top-level of a Pages document.

The <sl:prototype> elements define the captured pages and follow the structure of the document proper, which follows the <sl:section-prototypes> element, having child elements for stylesheet, headers, footers, drawables, text-storage, and thumbnails.

"Chapter 4, Working With Text Objects in Pages Documents," (page 43) describes in detail the usage of these elements and their attributes.

The hierarchy and structure of a Pages document is illustrated in "Figure 2-2."

Figure 1-2 The Pages document hierarchy and structure with associated child elements



The children and their usage in a Pages document are described in "Table 2-2":

Table 1-2 The Pages document structure with associated child elements

Elements	Description
<sl:version-history>	Lists the previous version of the document format.
<sl:publication-info>	Document-level globals, including user preferences.
<sl:metadata>	Spotlight metadata.
<sl:print-info>	Properties of a Cocoa NSPrintInfo object.

Elements	Description
<sl:section-prototypes>	A list of <sl:prototype> elements which contain data for captured pages, including the style sheet, drawables, headers, footers, the main body text, and thumbnails images.
<sf:stylesheet>	The document's main style sheet.
<sl:drawables>	Objects not in the main text flow, such as shapes, text boxes, and so on.
<sf:headers> and <sf:footers>	Lists the headers and footers used in the document.
<sf:text-storage>	The main body text of the document.
<sl:thumbnails>	Thumbnail images of the document pages.
<sl:window-configs>	Specifies the window size, position, and scroll-position, and the selection when the document was saved.

Defining Characteristics

In a high-level view of the Pages document structure, the first notable difference between a Keynote document and a Pages document is the <sl:section-prototypes> element, which is unique to Pages. The <sl:section-prototypes> contains a list of <sl:prototype> elements that mimic the structure of the main pages document, less those parts which pertain to the document as a whole.

A Pages <sf:stylesheet> element has exactly the same format as a Keynote <key:stylesheet>, except in Pages there is only one stylesheet for the main document, and one for each captured page set.

The <sl:drawables> element contains all the floating objects either for the page set, or the main document. These are grouped in a manner similar to the way they are grouped in Keynote—that is, into a masters-group and page-groups. Both the page-groups and masters-group have essentially the same structure. The masters-group are for drawables that are on the section masters, while the page-groups are the other drawables, grouped by the page on which they appear.

The <sl:masters-group> has a <sl:section-drawables> element for each section in the document or page set, indicated by its <sf:style> child element. Each <sl:section-drawables> element also has child elements in its foreground and background groups, which are in the same format as the Keynote drawables.

The <sl:drawables> element has <sl:page-group> child elements for each page with floating drawables.

The main document and each page set also have <sl:headers> and <sl:footers> elements that contain the headers and footers stored as arrays of text storages, and <sl:thumbnails> elements that contain the page thumbnail images in an ordered list, similar to the way that thumbnails are stored in Keynote.

The<sl:window-configs> element is used to specify the window size, location, and scroll position, and where the selection was when the document was saved.

Stylesheet Inheritance

In Pages, as contrasted with Keynote, there is no stylesheet inheritance. There is just one stylesheet per capture page set, and one for the rest of the document.

Document Elements

The next section describes some of the document elements in a Pages document. It is intended as a symbolic representation of the Pages document format rather than strictly valid XML.

Version History

The version-history is purely informational and is not interpreted by the application.

Publication-Info

The publication-info is defined in the following XML.

```
<sl:publication-info>
    <sl:kSFWPFootnoteFormatProperty>
      <sl:number sfa:number="0" sfa:type="i"/>
    </sl:kSFWPFootnoteFormatProperty>
    .
     .
    <sl:SLCreationDateProperty>
      <sl:date sfa:ID="NSCalendarDate-0" sl:val="2005-04-19T20:39:39 +0000"/>
    </sl:SLCreationDateProperty>
    .
     .
    <sl:SLLastModifiedDateProperty>
      <sl:date sfa:ID="NSDate-0" sl:val="2005-04-19T22:16:41 +0000"/>
    </sl:SLLastModifiedDateProperty>
     .
     .
</sl:publication-info>
```

Metadata

The metadata element is defined in the following XML:

```
<sf:metadata>
    <sf:comment>...</sf:comment>
    <sf:copyright>...</sf:copyright>
    <sf:keywords>...</sf:keywords>
    <sf:authors>...</sf:authors>
    <sf:title>...</sf:title>
</sf:metadata>
```

Print-Info

Some of the print-info elements that appear in the Pages XML are as follows:

```
<sl:slprint-info sfa:ID="SLPrintInfo-0" sl:page-width="612" sl:page-height="792"
 sl:page-scale="1">
```

```
     <sf:page-margins sfa:ID="SFWPMargins-0" sf:top="72"  sf:left="72"
sf:bottom="72"  sf:right="72" sf:header="36" sf:footer="43.200000762939453"/>
    <sl:print-info>
      <sl:NSJobDisposition>
      <sl:NSPaperSize>
      <sl:NSMustCollate>
      <sl:NSVerticalPagination>
      <sl:NSVerticallyCentered>
      <sl:NSPrintAllPages>
      <sl:NSCopies>
      <sl:NSBottomMargin>
      <sl:NSLeftMargin>
      <sl:NSTopMargin>
      <sl:NSLastPage>
      <sl:NSFirstPage>
      <sl:NSSavePath>
      <sl:NSOrientation>
    </sl:print-info>
</sl:slprint-info>
```

Section Prototypes

The section-prototypes XML describe the capture pages sets as follows:

```
<sl:section-prototypes>
    <sl:prototype sl:name="Text Page">
      <sf:stylesheet sfa:ID="SFSStylesheet-0">
        <sf:styles>
          <sf:paragraphstyle sfa:ID="SFWPParagraphStyle-0" sf:name="Free Form"
 sf:ident="paragraph-style-default">
                <sf:null/>
              </sf:listStyle>
              <sf:hidden/>
              <sf:underline/>
              <sf:underlineColor>
              <sf:word_strikethrough/>
              <sf:firstTopicNumber/>
            <sl:headers>...</sl: headers>
            <sl:footers>...</sl:footers>
            <sl:drawables>...</sl:drawables>
            <sl:text-storage>...</sl:text-storage>
            <sl:thumbnails>...</sl:thumbnails>
     </sl:prototype>
<sl:/section-prototypes>
```

Common Elements and Usage

Many elements in a Keynote or Pages document include a `sfa:ID` attribute, which assigns a unique name to the element by which it can be referred to elsewhere in the document, generally via a `sfa:IDREF` attribute.

A common use is when the same object appears in multiple places in the document, the first being the complete object with an `sfa:ID` attribute. In subsequent appearances, the object appears in an element of the same name and a `-ref` suffix with an `sfa:IDREF` attribute.

For example:

```
<key:object sfa:ID="Object-0">…<key:object/>
<key:object-ref sfa:IDREF="Object-0"/>
```

Both "Chapter 3, Deconstructing iWork Shapes and Drawable Objects," (page 25) and "Chapter 4, Working With Text Objects in Pages Documents," (page 43) discuss in detail the usage of these elements and attributes, with many XML examples for illustration.

Working With Document Bundles and Preferences

Both Pages documents and Keynote presentations are bundles. A bundle is a directory containing related resources and is treated as a single object (application, document, and so on) by Mac OS X. The term **bundle** refers both to the object and to the directory it represents.

Keynote and Pages document bundles contain a compressed XML file that describes the entire document, plus the media files that are used in the document, although certain media assets can be referenced externally.

For Keynote, some media files are actually kept in the theme bundles, which are part of the application. Likewise, media files belonging to Pages' templates are referenced from the template bundles to help reduce the size of user documents. A user can also choose whether to copy QuickTime movies into the document bundle.

Package Contents

To view the contents of a Pages or Keynote file, you can control-click it, and select Show Package Contents from the contextual menu that appears.

A Keynote document bundle contains an `index.apxl.gz` file containing the compressed XML for the document, a `Contents` folder that contains only the bundle `PkgInfo`, and a `Thumbs` folder, which stores a thumbnail image of each slides (seen in the slide sorter on the left column of the application). Media assets are also saved here.

Likewise, a Pages document bundle contains an `index.xml.gz file` containing the compressed XML for the document, a `Contents` folder that contains only the bundle `PkgInfo`, and a `Thumbs` folder, which stores thumbnail images of the pages. Media assets are also saved here.

The XML files use standard zlib compression, and may be decompressed by double-clicking the file, or with the gzip command-line tool. The applications can also recognize the files in their uncompressed form (without the `.gz` extension). Be aware that the compressed version is preferred if both exist in the document bundle.

Note that a Keynote 2.x index file is called `index.apxl`, while a Pages 1.x index file is specified simply as `index.xml`.

Decompressed Keynote Files

If you decompress a Keynote document's `index.apxl.gz` file and open it in a text editor, you'll see something similar to what is shown in "Listing 2-3."

Listing 1-3 Decompressed Keynote XML viewed in a text editor

```
 <key:presentation xmlns:sfa="http://developer.apple.com/namespaces/sfa"
xmlns:sf="http://developer.apple.com/namespaces/sf" xmlns:xsi="http://www.w3.org/
 2001/XMLSchema-instance" xmlns:key="http://developer.apple.com/namespaces/
keynote2" sfa:ID="BGShow-0" key:play-mode="interactive"  key:kiosk-slide-delay="5"
  key:kiosk-build-delay="2" key:mode="once" key:version="2004102100">
    <key:theme-list sfa:ID="NSMutableArray-0">
    <key:theme sfa:ID="BGTheme-0" key:name="Theme" key:decimal-tab=".">
      <key:size sfa:w="800" sfa:h="600"/>
      <key:stylesheet sfa:ID="SFSStylesheet-0">
        <sf:styles>
.
.
.
.
    <key:slide-list>
        <key:slide sfa:ID="BGSlide-6" key:depth="0">
    <key:ui-state>
    <key:version-history>
    <key:size>
</key:presentation>
```

In every Keynote document, the `<key:presentation>` element has a number of attributes, including four namespaces that are used throughout the document: (`sfa`, `sf`, `xsi`, and `key`), and a `key:play-mode`.

The `<key:presentation>` elements' children include `<key:theme-list>`, `<key:slide-list>` , `<key:ui-state>`, `<key:version-history>`, and `<key:size>`, as discussed earlier in this chapter.

In the example shown in "Listing 2-4," you'll see the partial structure of elements from the beginning of a `<key:theme-list>` element to the beginning of a child `<sf:styles>` element.

Listing 1-4 Keynote XML order from theme-list to styles

```
  <key:theme-list sfa:ID="NSMutableArray-0">
    <key:theme sfa:ID="BGTheme-0" key:name="Theme" key:decimal-tab=".">
      <key:size sfa:w="800" sfa:h="600"/>
      <key:stylesheet sfa:ID="SFSStylesheet-0">
        <sf:styles>
            .
            .
            .
```

Here the `<key:theme-list>` element has three children: a `<key:theme>` with a `key:name` attribute, a `<key:size>` with attributes specifying the width a `sfa:w` and the height `sfa:h`, and a `<key:stylesheet>`.

Decompressed Pages Documents

If you decompress a Pages document's `index.xml.gz` file and open it in a text editor, you'll see something similar to what is shown in "Listing 2-5" near the bottom of the file.

Listing 1-5 Decompressed Pages XML

```
<sf:text-storage sf:kind="body" sfa:ID="SFWPStorage-8">
  <sf:stylesheet-ref sfa:IDREF="SFSStylesheet-1"/>
  <sf:attachments/>
  <sf:footnotes>...</sf:footnotes>
  <sf:text-body>
    <sf:section sf:name="Chapter 1" sf:style="section-style-0">
      <sf:layout sf:style="layout-style-20">
        <sf:p sf:style="paragraph-style-32">Main body text.<sf:br/></sf:p>
        <sf:p sf:style="SFWPParagraphStyle-44">Second paragraph with a
<sf:span sf:style="SFWPCharacterStyle-5">character</sf:span> style and
a<sf:tab/>tab.<sf:br/></sf:p>
        <sf:p sf:style="paragraph-style-32">
          <sf:br/>
        </sf:p>
        .
        .
        .
      </sf:layout>
    </sf:section>
  </sf:text-body>
</sf:text-storage>
```

In this example, the `<sf:text-storage>` element has an attribute, `sf:kind`, which specifies that it pertains to the text in the document's main body.

The `<sf:text-storage>` element has four children: a `<sf:stylesheet-ref>`, which is a reference to a style sheet that appears earlier in the document, the `<sf:attachment>` and `<sf:footnotes>` elements, and a `<sf:text-body>` element.

The `<sf:text-body>` element has one or more child elements: an `<sf:section>`, which itself has one or more child `<sf:layout>` elements, which in turn has one or more `<sf:p>` elements, each of which has a `sf:style` attribute that specifies the identifiers of the sections, layouts, and paragraphs, respectively, of the first section, layout, and paragraph of the main body text.

Setting Preferences

Both Pages and Keynote have several preferences you can set that make it easier to work with their XML files:

```
defaults write com.apple.iWork.Pages SaveCompressionLevel 0
defaults write com.apple.iWork.Keynote SaveCompressionLevel 0
```

When the `SaveCompressionLevel` default is set to 0, the XML file will not be compressed. The normal value is 3.

```
defaults write com.apple.iWork.Pages FormatXML YES
defaults write com.apple.iWork.Keynote FormatXML YES
```

When the `FormatXML` default is set to `YES`, the XML will be written formatted for easier viewing and editing. The normal value is `NO`.

Deconstructing iWork Shapes and Drawable Objects

The first section of this chapter deconstructs the XML used in various iWork shapes, specifically Keynote shapes in a sample presentation. The XML structure for each shape is discussed, and the notable elements and attributes of these shapes are explained in detail.

An understanding of these sample shapes may be useful for iWork developers who want to modify or create their own shapes in Keynote presentations or Pages documents.

In the second section of this chapter, the basic structure of iWork drawable objects is defined. All of the shapes in the iWork XML shemas share a number of these structures, such as geometry, style, binary, path, image, and so on.

Variations in Keynote Shapes

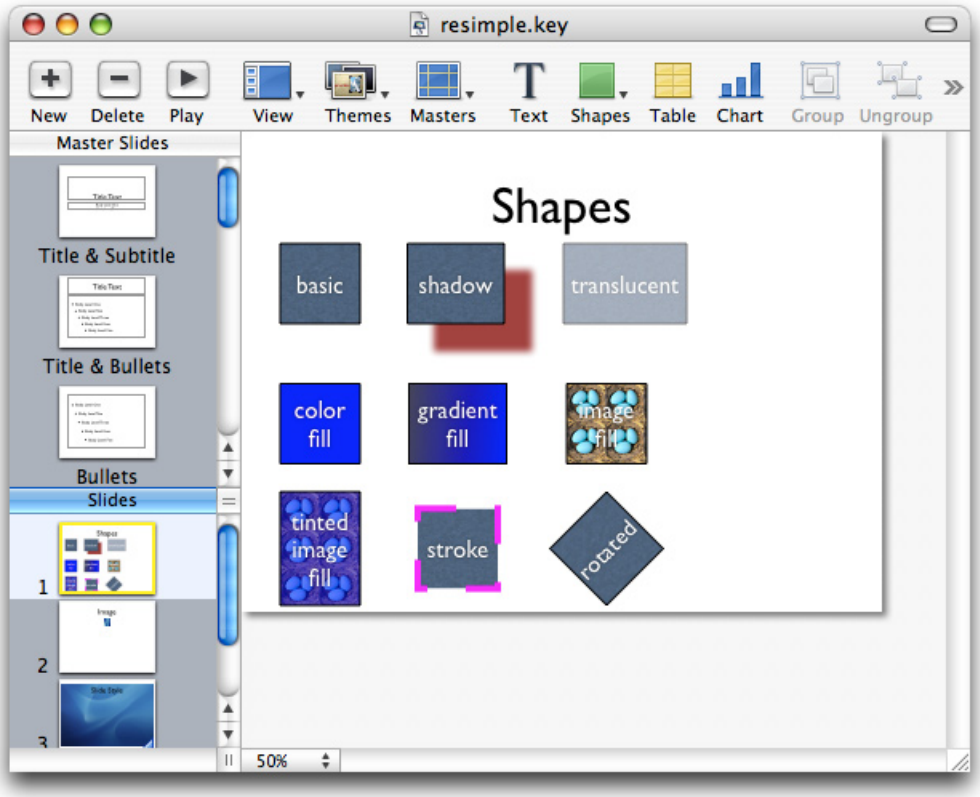
The different changeable parts of a Keynote shape comprise its graphic style. There are 12 different shapes available to users when opening a new Keynote presentation, including circles, squares, arrows, and so on.

Nine variations in shapes are shown in the sample Keynote presentation `resimple.key`, discussed in this chapter. The shapes are

- Basic
- Shadow
- Translucent
- Color fill
- Gradient fill
- Tinted fill
- Image fill
- Thick-color stroke
- Rotated default

The Keynote presentation `resimple.key` with the nine variations in shape is illustrated in “Figure 3-1.”

Figure 2-1 All nine shapes available in a Keynote presentation



Basic Shape

"Figure 3-2" illustrates the first and most basic shape available for a Keynote slide.

Figure 2-2 The basic shape

The basic shape in XML is defined in "Listing 3-1."

Listing 2-1 Basic shape XML

```
<sf:shape sfa:ID="SFDShapeInfo-2">
  <sf:geometry sfa:ID="SFDAffineGeometry-93">
   <sf:naturalSize sfa:w="100" sfa:h="100"/>
```

```
   <sf:size sfa:w="100" sfa:h="100"/>
   <sf:position sfa:x="48" sfa:y="140"/>
  </sf:geometry>
  <sf:style>
   <sf:graphic-style-ref sfa:IDREF="SFDGraphicStyle-62"/>
  </sf:style>
  <sf:path>
   <sf:bezier-path sfa:ID="SFDBezierPathSource-2">
    <sf:bezier sfa:ID="NSBezierPath-2" sfa:path="M 0 0 L 100  0 L 100 100 L 0
100 Z M 0 0" sfa:version="524"/>
   </sf:bezier-path>
  </sf:path>
  <sf:text sfa:ID="SFWPFrame-141">
   <sf:text-storage sfa:ID="SFWPStorage-477"  sf:kind="textbox"
sf:exclude-shapes="true" sf:exclude-tables="true"  sf:exclude-attachments="true">
    <sf:stylesheet-ref sfa:IDREF="SFSStylesheet-27"/>
    <sf:text-body>
     <sf:layout sf:style="SFWPLayoutStyle-189">
      <sf:p sf:style="SFWPParagraphStyle-390">basic</sf:p>
     </sf:layout>
    </sf:text-body>
  </sf:text-storage>
 </sf:text>
</sf:shape>
```

Defining Characteristics

The empty property map here indicates that all style properties of this shape are inherited from the master slide's prototype for shapes. This means that all the style properties are inherited from the master slide's stylesheet's style, with the identifier `"shapeStyleID"`.

The `<sf:geometry>` element consists of a natural size, a size, and a position. For shapes, natural size and size are the same, and determine how large the shape is on the canvas. The units of width and height are in pixels. The position is also in pixels, with (0,0) being the top left of the slide.

The `<sf:path>` element describes how the path is to be drawn. The format is the same as the format for the Keynote 1.x path elements. Note the `sfa:version` attribute, with a value of `"524"`.

Important: If you are a developer working with the Keynote XML file format specification, you should make sure that the `sfa:version` attribute is present and has a value of `"524"`, or the behavior of your application may become unpredictable.

The `<sf:text>` element describes the text that appears in the shape. Details of the text are beyond the scope of this chapter, but it is worth noting that you can change the text contents of a shape simply by changing what is specified in the `<sf:p>` element.

The Shadow Shape

"Figure 3-3" illustrates a simple shadow shape, another variation of the basic shape shown in "Figure 3-2."

Figure 2-3 The shadow shape

The elements for the shadow shape are defined, as follows in XML in "Listing 3-2":

Listing 2-2 Shadow shape XML

```
<sf:shape sfa:ID="SFDShapeInfo-3">
 <sf:geometry sfa:ID="SFDAffineGeometry-94">
  <sf:naturalSize sfa:w="100" sfa:h="100"/>
  <sf:size sfa:w="121" sfa:h="100"/>
  <sf:position sfa:x="207" sfa:y="140"/>
 </sf:geometry>
 <sf:style>
  <sf:graphic-style-ref sfa:IDREF="SFDGraphicStyle-59"/>
 </sf:style>
 <sf:path>
  <sf:bezier-path sfa:ID="SFDBezierPathSource-3">
   <sf:bezier sfa:ID="NSBezierPath-3" sfa:path="M 0 0 L 100 0 L 100 100 L 0 100
 Z M 0 0" sfa:version="524"/>
  </sf:bezier-path>
 </sf:path>
 <sf:text sfa:ID="SFWPFrame-142">
  <sf:text-storage sfa:ID="SFWPStorage-478"  sf:kind="textbox"
sf:exclude-shapes="true" sf:exclude-tables="true"  sf:exclude-attachments="true">
   <sf:stylesheet-ref sfa:IDREF="SFSStylesheet-27"/>
   <sf:text-body>
    <sf:layout sf:style="SFWPLayoutStyle-189">
     <sf:p sf:style="SFWPParagraphStyle-390">shadow</sf:p>
    </sf:layout>
   </sf:text-body>
  </sf:text-storage>
 </sf:text>
</sf:shape>
```

The shadow shape is similar to the previous example basic shape, except for changes to its geometry, style reference, and text. Otherwise, all the elements are identical to the basic shape. The shadow shape, of course, is in a different position, with a shadow, as well as with different text.

The shape's graphic style is defined as follows in XML in "Listing 3-3":

Listing 2-3 Graphic style XML

```
<sf:graphic-style sfa:ID="SFDGraphicStyle-59" sf:parent-ident="shapeStyleID">
 <sf:property-map>
```

```
  <sf:shadow>
   <sf:shadow sfa:ID="SFRShadow-9" sf:angle="45" sf:offset="50" sf:radius="10"
 sf:opacity="0.75">
    <sf:color xsi:type="sfa:calibrated-rgb-color-type" sfa:r="0.52822577953338623"
 sfa:g="0" sfa:b="0" sfa:a="1"/>
   </sf:shadow>
  </sf:shadow>
 </sf:property-map>
</sf:graphic-style>
```

Notably, the only attribute defined here is the `sf:shadow`; the other attributes are inherited from the master slide style with the identifier `"shapeStyleID"`.

The shadow property contains a `<sf:shadow>` element, which defines the angle, offset, radius and opacity of the shadow, as well as the color of the shadow. The `sf:angle` attribute corresponds to the angle in the application's inspector, although it is the opposite of what the inspector shows. In other words, the inspector shows 315, but the angle here is 45, which happens to be (360 - 315). The `sf:offset` attribute corresponds to the offset value in the inspector; the radius corresponds to the blur value in the inspector; and the opacity corresponds to the opacity value in the inspector.

The `<sf:color>` child defines the color of the shadow. There are many different ways to define colors, but this is one of the simplest. The `xsi:type` attribute of `"sfa:calibrated-rgb-color-type"` must be present so the application "knows" in what format the color is written. The other attributes are `sfa:r`, `sfa:g`, `sfa:b`, and `sfa:a`, which define the amount of red, blue and green in the color, and the opacity (alpha) of the color. These values can range from 0 to 1.

The Translucent Shape

"Figure 3-4" illustrates the translucent shape, and again, this shape only differs in its geometry, style and text from the previously discussed shapes.

Figure 2-4 The translucent shape

Because this shape does not differ significantly from previous shape elements, only its style is shown in the following XML, as illustrated in "Listing 3-4":

Listing 2-4 Translucent shape XML

```
<sf:graphic-style sfa:ID="SFDGraphicStyle-57" sf:parent-ident="shapeStyleID">
 <sf:property-map>
  <sf:opacity>
   <sf:number sfa:number="0.5150376" sfa:type="f"/>
  </sf:opacity>
```

```
 </sf:property-map>
</sf:graphic-style>
```

Defining Characteristics

Note that, again, the style inherits most of its values from the master style. It overrides `<sf:opacity>`. When the `<sf:opacity>` override occurs, it always has an `sfa:number` element with an `sfa:type` attribute of `sfa:f`, and an `sfa:number` attribute that varies from 0 to 1, with 1 being completely opaque, and 0 being completely transparent.

The Color Fill Shape

"Figure 3-5" illustrates the color fill shape.

Figure 2-5 The color fill shape

Here is the XML for the color fill shape as described in "Listing 3-5":

Listing 2-5 Color fill XML

```
<sf:shape sfa:ID="SFDShapeInfo-5">
 <sf:geometry sfa:ID="SFDAffineGeometry-96">
  <sf:naturalSize sfa:w="100" sfa:h="100"/>
  <sf:size sfa:w="100" sfa:h="100"/>
  <sf:position sfa:x="48" sfa:y="315"/>
 </sf:geometry>
 <sf:style>
  <sf:graphic-style-ref sfa:IDREF="SFDGraphicStyle-60"/>
 </sf:style>
 <sf:path>
  <sf:bezier-path sfa:ID="SFDBezierPathSource-5">
   <sf:bezier sfa:ID="NSBezierPath-5" sfa:path="M 0 0 L 100 0 L 100 100 L 0 100
 Z M 0 0" sfa:version="524"/>
   </sf:bezier-path>
 </sf:path>
 <sf:text sfa:ID="SFWPFrame-144">
  <sf:text-storage sfa:ID="SFWPStorage-480"  sf:kind="textbox"
sf:exclude-shapes="true" sf:exclude-tables="true"  sf:exclude-attachments="true">
   <sf:stylesheet-ref sfa:IDREF="SFSStylesheet-27"/>
   <sf:text-body>
    <sf:layout sf:style="SFWPLayoutStyle-189">
     <sf:p sf:style="SFWPParagraphStyle-390">color<sf:br/></sf:p>
     <sf:p sf:style="SFWPParagraphStyle-390">fill</sf:p>
```

```
    </sf:layout>
   </sf:text-body>
  </sf:text-storage>
 </sf:text>
</sf:shape>
```

In this shape, both the geometry and the text are different from the previously discussed shapes. For text, instead of a single <sf:p> element in the <sf:layout> element, there are two, and one has an <sf:br/> element, which indicates a line break.

The different fill is, of course, encoded in the style:

```
<sf:graphic-style sfa:ID="SFDGraphicStyle-60" sf:parent-ident="shapeStyleID">
 <sf:property-map>
  <sf:fill>
   <sf:color xsi:type="sfa:calibrated-rgb-color-type" sfa:r="0" sfa:g="0"
sfa:b="1" sfa:a="1"/>
  </sf:fill>
 </sf:property-map>
</sf:graphic-style>
```

As with other shapes, the color fill shape inherits most of its properties from the master style. However, the fill property is overridden, and its value is an <sf:color>. The <sf:color> element here takes the same format as the <sf:color> element in the shadow example.

The Gradient Fill Shape

"Figure 3-6" illustrates a gradient fill shape, and again, the only notable change in this shape is the style.

Figure 2-6 The gradient fill shape

The XML for the gradient fill shape is as follows in "Listing 3-6":

Listing 2-6 Gradient fill shape XML

```
<sf:graphic-style sfa:ID="SFDGraphicStyle-64" sf:parent-ident="shapeStyleID">
 <sf:property-map>
  <sf:fill>
   <sf:angle-gradient sfa:ID="SFRAngleGradient-0" sf:opacity="1" sf:type="linear"
 sf:angle="3.0717794895172119">
    <sf:stops sfa:ID="NSMutableArray-81">
     <sf:gradient-stop sfa:ID="SFRGradientStop-0" sf:fraction="0">
```

```
      <sf:color xsi:type="sfa:calibrated-rgb-color-type" sfa:r="0" sfa:g="0"
sfa:b="1" sfa:a="1"/>
     </sf:gradient-stop>
     <sf:gradient-stop sfa:ID="SFRGradientStop-1" sf:fraction="1">
      <sf:color xsi:type="sfa:calibrated-rgb-color-type"
sfa:r="0.27450981736183167" sfa:g="0.27450981736183167"
sfa:b="0.34509804844856262" sfa:a="1"/>
     </sf:gradient-stop>
    </sf:stops>
   </sf:angle-gradient>
  </sf:fill>
 </sf:property-map>
</sf:graphic-style>
```

Defining Properties and Elements

The `sf:fill` property here is similar to the `sf:fill` property in the color fill example, except that instead of having a color, it has an `<sf:angle-gradient>`. The `xsi:type` attribute of the angle-gradient should always be `"linear"` . (It is provided for forward compatibility.) The angle is the amount that the gradient differs from 180. Opacity refers to how opaque the gradient is.

The `<sf:angle-gradient>` element contains an `<sf:stops>` element, which must contain two or more stops. Each stop has a `sf:fraction` attribute, which describes how far along the gradient the stop occurs, with 0 being the beginning of the gradient and 1 being the end. The value of this attribute should always be larger for any stop than for the previous stop within the gradient. Each `<sf:gradient-stop>` contains a color, which is of the same format as the color for shadows or color fills.

The Image Fill Shape

"Figure 3-7" illustrates an image fill, whose only notable difference from the previous shapes is, again, its style.

Figure 2-7 The image fill shape

The XML for the image fill shape is shown in "Listing 3-7":

Listing 2-7 Image fill XML

```
<sf:graphic-style sfa:ID="SFDGraphicStyle-61" sf:parent-ident="shapeStyleID">
 <sf:property-map>
  <sf:fill>
```

```
   <sf:textured-fill sfa:ID="SFDTexturedImageFill-28" sf:technique="tile"
xsi:type="textured-fill">
    <sf:image sfa:ID="SFRImageBinary-38">
     <sf:size sfa:w="48" sfa:h="48"/>
     <sf:data sfa:ID="SFEData-39" sf:path="Nest.tif"  sf:displayname="Nest.tif"
 sf:size="22520" sf:hfs-type="0"  sf:checksum="4bf7bacf"  sfa:version="1"/>
    </sf:image>
   </sf:textured-fill>
  </sf:fill>
 </sf:property-map>
</sf:graphic-style>
```

Defining Properties and Attributes

The `<sf:fill>` property, as shown in the above XML, has an `"textured-image-fill"` child. The `sf:technique` attribute describes how the image is drawn to fill the space: it can tile, scale to fit, scale to fill, or stretch the image, or leave the image as its original size.

The `"image-binary"` child defines the actual image. The `<sf:size>` child of the image-binary child defines the dimensions, in pixels, of the unscaled image. The `sf:path` is relative to the document bundle, and the `sf:displayname` is the name as it appears in the inspector. The `sf:hfs-type` attribute describes the type of the file as far as the Finder is concerned, but may be omitted. The `sf:size` and `sf:checksum` attributes are used to determine if the image has changed, and may be omitted. In fact, it is preferable to omit these attributes rather than define them incorrectly. The `sf:version` attribute should always be `"1"`.

The Tinted Image Fill

"Figure 3-8" illustrates a tinted image fill, which is similar to the image fill in the above example.

Figure 2-8 The tinted image fill

The XML for the tinted image fill is "Listing 3-8":

Listing 2-8 Tinted image fill XML

```
<sf:graphic-style sfa:ID="SFDGraphicStyle-63" sf:parent-ident="shapeStyleID">
 <sf:property-map>
```

```
  <sf:fill>
   <sf:textured-fill sfa:ID="SFDTexturedImageFill-29"  sf:technique="tile"
xsi:type="textured-fill">
    <sf:color xsi:type="sfa:calibrated-rgb-color-type" sfa:r="0" sfa:g="0"
sfa:b="1" sfa:a="0.5"/>
    <sf:image-ref sfa:IDREF="SFRImageBinary-38"/>
   </sf:textured-fill>
  </sf:fill>
</sf:property-map>
```

While the tinted image fill shape is similar in many respects to the image fill example, there are several notable differences:

- The `<sf:textured-fill>` element has a `sf:color` attribute, similar to the `sf:color` attribute described in the fill and shadow examples.
- The `<sf:image>`, instead of being a full-fledged image element, is an `image-ref` that refers to the same image used for the previous example. This allows the document to use the same image multiple times without having to write it over and over again, thus keeping the document simpler, smaller, and more consistent.

The Thick-Colored Stroke

"Figure 3-9" illustrates a thick-colored stroke.

Figure 2-9 The thick-colored stroke

Its style is defined in the XML in "Listing 3-9":

Listing 2-9 Thick-colored stroke XML

```
<sf:graphic-style sfa:ID="SFDGraphicStyle-58" sf:parent-ident="shapeStyleID">
 <sf:property-map>
  <sf:stroke>
   <sf:stroke sfa:ID="SFRStroke-123" sf:miter-limit="4" sf:width="8" sf:cap="butt"
 sf:join="miter">
    <sf:color xsi:type="sfa:calibrated-rgb-color-type" sfa:r="1" sfa:g="0"
sfa:b="1" sfa:a="1"/>
    <sf:pattern sfa:ID="SFRStrokePattern-123" sf:phase="0" sf:type="pattern">
     <sf:pattern>
      <sf:element sf:val="6"/>
      <sf:element sf:val="6"/>
     </sf:pattern>
```

```
    </sf:pattern>
   </sf:stroke>
  </sf:stroke>
 </sf:property-map>
</sf:graphic-style>
```

Notable Properties and Child Elements

The stroke property has a `<sf:stroke>` child, with a `sf:width` attribute that defines how many pixels wide the stroke is. The `sf:miter-limit` and `sf:cap` attributes define how the stroke joins to itself, but there is no user interface exposed for editing these.

Important: Developers should avoid using different values for these attributes than the ones specified in the above example.

The `<sf:color>` child defines the color of the stroke, and is of the same format as the previous color examples.

The `<sf:pattern>` element contains a pattern child which has two, four or six element children. The first element child of each pair describes the length of a piece of line that is drawn, and the next element describes the length of an empty space.

The Rotated Default Shape

"Figure 3-10" illustrates a default shape that has been rotated 45 degrees.

Figure 2-10 The rotated shape

Not surprisingly, there is a change in the geometry, as shown in the following XML in "Listing 3-10":

Listing 2-10 Rotated default shape XML

```
<sf:shape sfa:ID="SFDShapeInfo-10">
 <sf:geometry sfa:ID="SFDAffineGeometry-101" sf:angle="45">
  <sf:naturalSize sfa:w="100" sfa:h="100"/>
  <sf:size sfa:w="100" sfa:h="100"/>
  <sf:position sfa:x="385.289306640625" sfa:y="450.289306640625"/>
```

```
 </sf:geometry>
 <sf:style>
  <sf:graphic-style-ref sfa:IDREF="SFDGraphicStyle-56"/>
 </sf:style>
 <sf:path>
  <sf:bezier-path sfa:ID="SFDBezierPathSource-10">
   <sf:bezier sfa:ID="NSBezierPath-10" sfa:path="M 0 0 L 100 0 L 100 100 L 0
100 Z M 0 0" sfa:version="524"/>
  </sf:bezier-path>
 </sf:path>
 <sf:text sfa:ID="SFWPFrame-149">
  <sf:text-storage sfa:ID="SFWPStorage-573"  sf:kind="textbox"
sf:exclude-shapes="true" sf:exclude-tables="true"  sf:exclude-attachments="true">
   <sf:stylesheet-ref sfa:IDREF="SFSStylesheet-27"/>
   <sf:text-body>
    <sf:layout sf:style="SFWPLayoutStyle-189">
     <sf:p sf:style="SFWPParagraphStyle-390">rotated</sf:p>
    </sf:layout>
   </sf:text-body>
  </sf:text-storage>
 </sf:text>
</sf:shape>
```

In this example, note the addition of the `sf:angle` attribute on the `<sf:geometry>` element. This defines the amount of rotation, and is identical to the value the user sees in the Keynote Inspector.

The Overriden Slide Style

"Figure 3-11" illustrates the overriden slide style, which is unique to Keynote.

Figure 2-11 The overriden slide style

The XML looks like this in"Listing 3-11":

Listing 2-11 The overriden slide style XML

```
<sf:slide-style sfa:ID="BGSlideStyle-16" sfa:sfclass="" sf:ident="slideStyleID"
 sf:parent-ident="slideStyleID">
 <sf:property-map>
  <sf:bodyPlaceholderVisibility>
   <sf:number sfa:number="1" sfa:type="c"/>
  </sf:bodyPlaceholderVisibility>
  <sf:slideNumberPlaceholderVisibility>
   <sf:number sfa:number="1" sfa:type="c"/>
  </sf:slideNumberPlaceholderVisibility>
  <sf:fill>
   <sf:textured-fill sfa:ID="SFDTexturedImageFill-27" sf:technique="fit"
xsi:type="textured-fill">
    <sf:image sfa:ID="SFRImageBinary-27">
     <sf:size sfa:w="1600" sfa:h="1024"/>
    <sf:data sfa:ID="SFEData-28" sf:path="Aqua  Blue.jpg"  sf:displayname="Aqua
 Blue.jpg" sf:size="637181"  sf:hfs-type="0"  sf:checksum="e5d6c9c9"
sfa:version="1"/>
     </sf:image>
    </sf:textured-fill>
   </sf:fill>
   <sf:transition>
    <sf:transition-attributes sfa:ID="BGTransitionAttributes-11"
key:type="apple:3D-cube">
     <key:animationAuto>
      <key:number sfa:number="1" sfa:type="c"/>
    </key:animationAuto>
    <key:animationDelay>
      <key:decimal-number sfa:number="1.5" sfa:type="d"/>
    </key:animationDelay>
    <key:direction>
     <key:number sfa:number="11" sfa:type="i"/>
    </key:direction>
   </sf:transition-attributes>
  </sf:transition>
 </sf:property-map>
</sf:slide-style>
```

Defining Characteristics and Properties

This slide style inherits from a master with only the title placeholder visible. However, it also makes the body and slide number placeholders visible. This is accomplished with the `<sf:bodyPlaceholderVisibility>` and `<sf:slideNumberPlaceholderVisibility>` elements, each of which has a child element that is an `<sf:number>` element with an `sfa:type` of `"c"` and an `sfa:number` of 0 or 1. Both elements have a value of 1, meaning that the placeholders are visible. To make a placeholder invisible, if it is visible on the master, use a value of 0.

The slide also has an image fill. This is encoded in the `<sf:fill>` property. The `<sf:fill>` property for a slide style is identical to the fill property for a graphic style: it can be a color, gradient, or image fill (with or without tinting.)

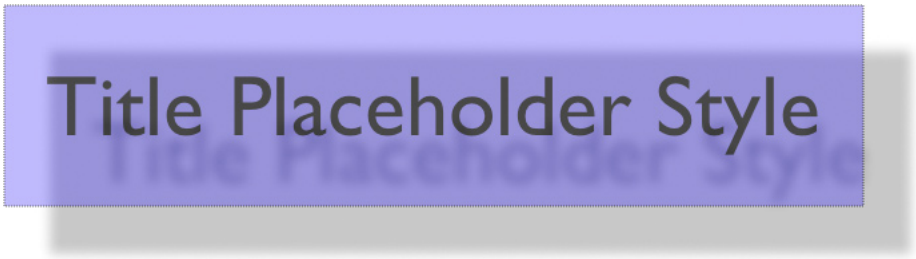
The `<sf:transition>` property defines the transition between this slide and the next. The `sfa:type` attribute is a string that encodes the name of the transition. The `<key:animationAuto>` property is a number similar to the number for placeholder visibility, with a value of 0 if the user must take some action to show the next slide, and 1 if the next slide is shown automatically. If the next slide is shown

automatically, the <key:animationDelay> property encodes the length of the delay from when the slide starts (or the last build completes) and the transition begins. The key:direction attribute has a different meaning for different transitions.

The Title Placeholder Style

"Figure 3-11" illustrates a title placeholder style (unique to Keynote) with customized position, stroke, shadow, and fill.

Figure 2-12 The title placeholder style



The title placeholder style is defined in "Listing 3-12":

Listing 2-12 The title placeholder style XML

```
<sf:placeholder-style sfa:ID="BGPlaceholderStyle-114" sfa:sfclass=""
sf:ident="titlePlaceholderStyleID" sf:parent-ident="titlePlaceholderStyleID">
 <sf:property-map>
  <sf:fill>
   <sf:color xsi:type="sfa:calibrated-rgb-color-type" sfa:r="0" sfa:g="0"
sfa:b="1" sfa:a="0.5"/>
  </sf:fill>
  <sf:stroke>
   <sf:stroke sfa:ID="SFRStroke-124" sf:miter-limit="4" sf:width="1" sf:cap="butt"
 sf:join="miter">
    <sf:color xsi:type="sfa:calibrated-white-color-type" sfa:w="0" sfa:a="1"/>
    <sf:pattern sfa:ID="SFRStrokePattern-124" sf:phase="0" sf:type="pattern">
     <sf:pattern>
      <sf:element sf:val="1"/>
      <sf:element sf:val="1"/>
     </sf:pattern>
    </sf:pattern>
   </sf:stroke>
  </sf:stroke>
  <sf:geometry>
   <sf:geometry sfa:ID="SFDAffineGeometry-103" sf:sizesLocked="true">
    <sf:naturalSize sfa:w="644" sfa:h="150"/>
    <sf:size sfa:w="644" sfa:h="150"/>
    <sf:position sfa:x="77" sfa:y="224"/>
    </sf:geometry>
   </sf:geometry>
   <sf:opacity>
    <sf:number sfa:number="0.7030075" sfa:type="f"/>
   </sf:opacity>
```

```
   <sf:shadow>
    <sf:shadow sfa:ID="SFRShadow-10" sf:angle="45" sf:offset="50" sf:radius="10"
 sf:opacity="0.75">
     <sf:color xsi:type="sfa:calibrated-rgb-color-type" sfa:r="0" sfa:g="0"
sfa:b="0" sfa:a="1"/>
    </sf:shadow>
   </sf:shadow>
  </sf:property-map>
</sf:placeholder-style>
```

The stroke, shadow, and fill are identical in form to the same properties for graphic styles.

Notably, the geometry for the placeholder is not stored in the `<sf:placeholder>` element, but rather, in its style. The form of this element is the same as the form of the geometry element for the shape or image, but lives in the style.

Defining the Structure of Drawable Objects

In this section, the basic structure of iWork drawable objects is discussed. All shapes in the iWork XML shemas share some of these structures.

Drawable Object Geometry

"Listing 3-13" defines the basic geometry of an iWork drawable object in XML, as contained by the `<sf:geometry>` element.

Listing 2-13 The basic geometry of a drawable object

```
<sf:geometry sfa:ID="SFDAffineGeometry-0" sf:sizesLocked="true" sf:angle="315">
    <sf:naturalSize sfa:w="100.0" sfa:h="0.0"/>
    <sf:size sfa:w="100.0" sfa:h="0.0"/>
    <sf:position sfa:x="100.0" sfa:y="100.0"/>
</sf:geometry>
```

The `sfa:ID` attribute is a unique string. The `sf:sizesLocked` attribute specifies whether the drawable object is locked with the `"true"` identifier. The `sf:angle` attribute specifies the angle of rotation as a float with a range of 0-360. The child element in this example (`<sf:naturalSize>`) is the unscaled size of the object, and for all but images should be the same as `<sf:size>`. The `<sf:size>` is the scaled size of the object. The `<sf:position>` defines the position on the page or slide.

Drawable Object Style

"Listing 3-14" defines the basic style of an iWork drawable object in XML, as contained by the `<sf:style>` element.

Listing 2-14 The basic style of a drawable object

```
<sf:style>
    <sf:graphic-style-ref sfa:IDREF="SFDGraphicStyle-33"/>
```

```
</sf:style>
```

The child element <sf:graphic-style-ref> references the graphic style.

Drawable Object Path

"Listing 3-15" defines the shape of a path of an iWork drawable object in XML, as contained by the <sf:path> element.

Listing 2-15 The basic path of a drawable object

```
<sf:path>
    <sf:bezier-path sfa:ID="SFDBezierPathSource-3">
        <sf:bezier sfa:ID="NSBezierPath-9" sfa:path="M 0 50 L 50 100 L 100 50
L 50 0 Z M 0 50" sfa:version="524"/>
    </sf:bezier-path>
</sf:path>
```

The child element <sf:bezier-path> describes how the path is to be drawn. The format is the same as the format for the Keynote 1.x path elements. The sfa:version attribute must have a value of "524".

Important: If you are a developer working with the Keynote XML file format specification, you should make sure that the sfa:version attribute is present and has a value of "524", or the behavior of your application may become unpredictable.

Drawable Object Binary

"Listing 3-16" defines the binary of an iWork drawable object for an image, such as a JPEG, TIFF, PDF, or EPS in XML, as contained by the <sf:binary> element.

Listing 2-16 The basic binary of a drawable object

```
<sf:binary sfa:ID="SFRImageBinary-2">
    <sf:size sfa:w="77" sfa:h="46"/>
    <sf:data sfa:ID="SFEData-2" sf:path="testA.tiff"  sf:displayname="testA.tiff"
  sf:size="14362" sf:hfs-type="0" sf:checksum="707facb4"  sfa:version="1"/>
</sf:binary>
```

The child element <sf:size> is the size of the original image, with the attributes sfa:w and sfa:h having float values that define the width and height. The <sf:data> element describes the image file with the following attributes:

- sf:path, which defines the path, either as a bundle or absolute path name to the image file. (A bundle-relative path might be renamed to avoid collisions.)
- sf:displayname, which displays the name of the image shown in the metrics inspector.
- sf:size, which specifies the optional size of the file in bytes, and is used to determine if the file has changed.

- sf:checksum, which is the checksum of the file data, and is used to determine if the file has changed.
- sfa:version, which must have a value of "1".

Drawable Object Line Shape

"Listing 3-17" defines the basic line shape of an iWork drawable object in XML, as contained by the <sf:line> element.

Listing 2-17 The basic line of a drawable object

```
<sf:line sfa:ID="SFDLineInfo-0">
    <sf:geometry sfa:ID="SFDAffineGeometry-0" sf:sizesLocked="true"
sf:angle="315">
        <sf:naturalSize sfa:w="100.0" sfa:h="0.0"/>
        <sf:size sfa:w="100.0" sfa:h="0.0"/>
        <sf:position sfa:x="100.0" sfa:y="100.0"/>
    </sf:geometry>
    <sf:style>
        <sf:graphic-style-ref sfa:IDREF="SFDGraphicStyle-0"/>
    </sf:style>
    <sf:head sfa:x="100.0" sfa:y="100.0"/>
    <sf:tail sfa:x="100.0" sfa:y="100.0"/>
</sf:line>
```

The child elements in this example are <sf:geometry>, <sf:style>, <sf:head>, and <sf:tail>. The <sf:head> defines the position of the head of the line, with attributes sfa:x and sfa:y having the floating point coordinates. The <sf:tail> defines the position of the tail of the line, with attributes sfa:x and sfa:y having the floating point coordinates.

Drawable Object Basic Shape

"Listing 3-18" shows an example of the basic shape of an iWork drawable object in XML, as contained by the <sf:drawable-shape> element.

Listing 2-18 The basic shape of a drawable object

```
<sf:drawable-shape sfa:ID="SLShapeInfo-2" sfa:sfclass="shape"
sl:max-attachment-scale-x="0.70710676908493042"
sl:max-attachment-scale-y="0.70710676908493042">
  <sf:geometry sfa:ID="SFDAffineGeometry-11">
    <sf:naturalSize sfa:w="100" sfa:h="100"/>
    <sf:size sfa:w="100" sfa:h="100"/>
    <sf:position sfa:x="72" sfa:y="274"/>
  </sf:geometry>
  <sf:style>
    <sf:graphic-style-ref sfa:IDREF="SFDGraphicStyle-38"/>
  </sf:style>
  <sf:path>
    <sf:bezier-path sfa:ID="SFDBezierPathSource-2">
       <sf:bezier sfa:ID="NSBezierPath-8" sfa:path="M  85.355339050292969
14.644660949707031 C 104.88155364990234 34.170875549316406 104.88155364990234
```

```
 65.829124450683594 85.355339050292969 85.355339050292969 C 65.829124450683594
  104.88155364990234 34.170875549316406 104.88155364990234 14.644660949707031
 85.355339050292969 C -4.8815536499023438 65.829124450683594 -4.8815536499023438
  34.170875549316406 14.644660949707031 14.644660949707031 C 34.170875549316406
  -4.8815536499023438 65.829124450683594 -4.8815536499023438 85.355339050292969
  14.644660949707031" sfa:version="524"/>
    </sf:bezier-path>
  </sf:path>
  <sf:text sfa:ID="SFWPFrame-14">
    <sf:text-storage sfa:ID="SFWPStorage-22" sf:kind="textbox"
sf:exclude-tables="true">
      <sf:stylesheet-ref sfa:IDREF="SFSStylesheet-1"/>
      <sf:text-body>
        <sf:layout sf:style="graphic-shape-layout-style-default">
          <sf:p sf:style="paragraph-style-default"/>
        </sf:layout>
      </sf:text-body>
    </sf:text-storage>
  </sf:text>
</sf:drawable-shape>
```

The `sfa:class` attribute always has a value of `"shape"`.

The child elements in this example are `<sf:geometry>`, `<sf:style>`, `<sf:path>`, and `<sf:text>`. The `<sf:text>` element defines the text inside of the shape.

Text objects are discussed in detail in "Chapter 4, Working With Text Objects in Pages Documents," (page 43).

Drawable Object Images

"Listing 3-19" defines the basic images of an iWork drawable object in XML, as contained by the `<sf:image>` element.

Listing 2-19 The basic images of a drawable object

```
<sf:image sfa:ID="SFDImageInfo-1" sf:image-placeholder="true">
  <sf:geometry sfa:ID="SFDAffineGeometry-19" sf:aspectRatioLocked="true">
    <sf:naturalSize sfa:w="77" sfa:h="46"/>
    <sf:size sfa:w="77" sfa:h="46"/>
    <sf:position sfa:x="72" sfa:y="72"/>
  </sf:geometry>
  <sf:style>
    <sf:graphic-style-ref sfa:IDREF="SFDGraphicStyle-33"/>
  </sf:style>
  <sf:binary sfa:ID="SFRImageBinary-2">
    <sf:size sfa:w="77" sfa:h="46"/>
    <sf:data sfa:ID="SFEData-2" sf:path="testA.tiff"  sf:displayname="testA.tiff"
  sf:size="14362" sf:hfs-type="0" sf:checksum="707facb4"  sfa:version="1"/>
  </sf:binary>
</sf:image>
```

The child elements in this example are `<sf:geometry>`, `<sf:style>`, and `<sf:binary>`. The `sf:image-placeholder` attribute has a value of `"true"` if the image is a placeholder; otherwise, this attribute is omitted.

Working With Text Objects in Pages Documents

This chapter works through a series of examples in Pages documents, describing how text objects in each example are stored and laid out in XML. Understanding how Pages documents store and lay out text is important if you want to customize and design your own page layouts, or modify the existing XML of your documents.

How Pages Stores and Lays Out Various Text Objects

The chapter discusses the following group of examples used in Pages documents to lay out various text objects:

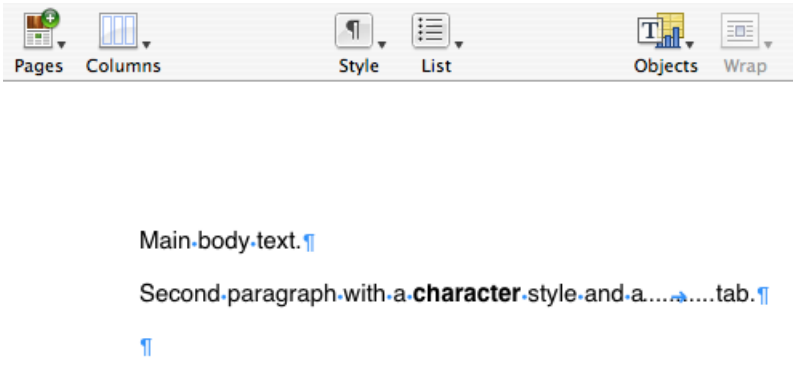
- "Main Body Text Objects With Character Styles" (page 43) examines a page layout comprised of main body text, along with a bolded character style in its second paragraph, and a tab.
- "Lists Text Object" (page 45) discusses the elements and attributes of a list of text items in a hierarchical outline view.
- "Multiple Column Text Objects" (page 47) discusses a page layout that is comprised of multiple columns.
- "Text Objects With Date-Time Fields and Hyperlinks" (page 48) discusses how you can work with a page layout comprised of a date-time field, page numbers, bookmarks, and hypertext links.
- "A Footnote Text Object" (page 50) looks at a text object with two footnotes and their corresponding superscripts.
- "A Text Object With an Image and a Drawable Attachment" (page 51) explains how you can work with a text object with an image and a drawable attachment.

The examples are intended to illustrate how Pages documents store text objects in XML and how they are laid out.

Main Body Text Objects With Character Styles

"Figure 4-1" illustrates the output for a Pages text storage object with a page layout comprised of a main body text and a bolded character style in a second paragraph, followed by a tab. "Listing 4-1" describes the XML for this text storage object.

Figure 3-1 Example output for the main body text and a bolded character style, followed a tab



Listing 3-1 The XML for the main body text and a second paragraph with a bolded character style

```
<sf:text-storage sf:kind="body" sfa:ID="SFWPStorage-8">
  <sf:stylesheet-ref sfa:IDREF="SFSStylesheet-1"/>
  <sf:text-body>
    <sf:section sf:name="Chapter 1" sf:style="section-style-0">
      <sf:layout sf:style="layout-style-20">
        <sf:p sf:style="paragraph-style-32">Main body text.<sf:br/></sf:p>
        <sf:p sf:style="SFWPParagraphStyle-44">Second paragraph with a
<sf:span sf:style="SFWPCharacterStyle-5">character</sf:span> style and
a<sf:tab/>tab.<sf:br/></sf:p>
        <sf:p sf:style="paragraph-style-32">
          <sf:br/>
        </sf:p>
      </sf:layout>
    </sf:section>
  </sf:text-body>
</sf:text-storage>
```

Defining Characteristics

The `<sf:text-storage>` element has one required attribute other than the `sfa:ID` attribute: the `sf:kind`, which indicates the usage of the text. The value must be one of `"body"`, `"header"`, `"footnote"`, `"textbox"`, `"note"`, or `"cell"`. The text storage `sf:kind` attribute must have a value corresponding to its use in the document. For example, the text storage used for a header must have a `sf:kind="header"` in order to avoid undefined or unpredictable types of behavior.

A `<sf:text-storage>` element must have a `<sf:stylesheet-ref>` child element that refers to the document or prototype stylesheet and may have a child `<sf:attachment>` element. Body text storages can also have a `<sf:footnotes>` child element.

A `<sf:text-storage>` element must have a `<sf:text-body>` child element. The next immediate child of the `<sf:text-storage>` element depends on the text storage kind.

The child elements of a `<sf:text-storage>` with an `sf:kind` of `"body"` must follow the strict hierarchy of `<sf:section>` elements, containing `<sf:layout>` elements that contain `<sf:p>` elements, which indicate the section, layout style, and paragraph style, respectively, of the contained mixed content.

The child elements of a <sf:text-storage> with an sf:kind of "textbox", "cell", or "note" must follow the strict hierarchy of <sf:layout> elements containing <sf:p> elements, which indicate the layout style, and paragraph style, respectively, of the contained mixed content.

The child elements of a <sf:text-storage> with an sf:kind of "header" or "footnote" elements contain <sf:p> elements, which indicate the paragraph style of the contained mixed content.

Note: The <sf:p> element does not itself indicate the break character which ends the paragraph and must be included immediately prior to the close of the paragraph element (except on the last paragraph of the text storage).

Style runs at the sub-paragraph level are enclosed in <sf:span> elements, and reference character styles. Note that spans are used only for sub-paragraph style runs and define only the variations from the enclosing paragraph style. Any style that applies to all of the text in a paragraph should be defined by the paragraph style. Note that <sf:span> elements do not nest.

The standard paragraph break is indicated by a <sf:br/> element. Other paragraph breaking elements are <sf:pgbr/> for page breaks, <sf:sectbrbr/> for section breaks, <sf:layoutbr/> for layout breaks, <sf:contbr/> for container breaks, and <sf:footnotebr/> for footnote breaks.

Style References

All styles in an iWork document have up to three attributes that can be used to identify the style. Like many other elements, all style elements have an sfa:ID attribute with a unique string as a value. Many style elements also have an sf:ident attribute with a value that is unique for all style elements of that type. Some style elements also have an sf:name attribute with a user-visible string as the value.

Text styles are referred to elsewhere in the document by a sf:style attribute the value of which is either the style's sf:ident, or its sfa:ID.

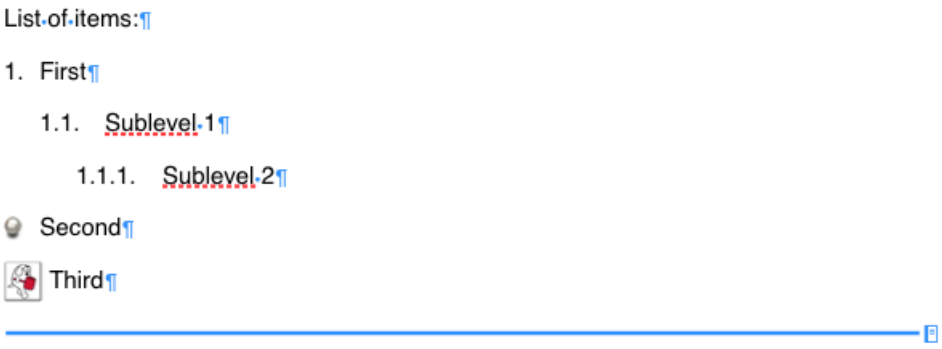
The <sf:section> element must have an <sf:name> attribute (which is not exposed to the user), and an sf:style attribute referencing a section style.

The <sf:layout> and <sf:p> elements reference layout and paragraph styles, respectively.

Lists Text Object

"Figure 4-2" illustrates the output for a Pages text object with a page layout comprised of a list of items in an outline view. "Listing 4-2" describes the XML for text in outline form.

Figure 3-2 Example output for a list of items in outline form



Listing 3-2 The XML for an item list in outline form

```
<sf:text-storage sf:kind="body" sfa:ID="SFWPStorage-8">
    <sf:stylesheet-ref sfa:IDREF="SFSStylesheet-1"/>
    <sf:text-body>
        <sf:section sf:name="Chapter 1" sf:style="section-style-0">
        <sf:layout sf:style="layout-style-20">
      <sf:p sf:style="paragraph-style-32">List of items:<sf:br/></sf:p>
      <sf:p sf:style="SFWPParagraphStyle-45"
sf:restart-list="true">First<sf:br/></sf:p>
      <sf:p sf:style="SFWPParagraphStyle-45" sf:list-level="1">Sublevel
1<sf:br/></sf:p>
      <sf:p sf:style="SFWPParagraphStyle-45" sf:list-level="2">Sublevel
2<sf:br/></sf:p>
      <sf:p sf:style="SFWPParagraphStyle-46">Second<sf:br/></sf:p>
      <sf:p sf:style="SFWPParagraphStyle-43">Third<sf:br/></sf:p>
      <sf:p sf:style="paragraph-style-32">
        <sf:sectbr/>
            </sf:p>
        </sf:layout>
        </sf:section>
    </sf:text-body>
</sf:text-storage>
```

Defining Characteristics

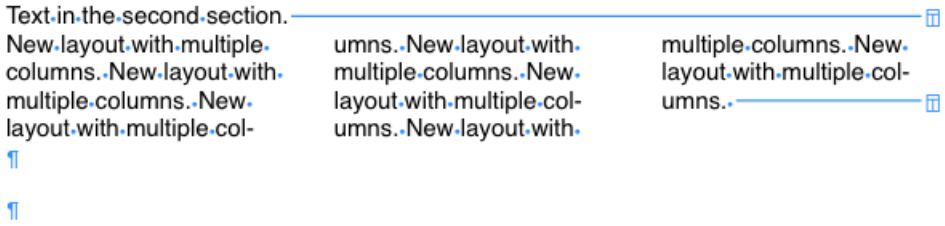
The fact that a paragraph is an element in a list is indicated by an `sf:list-level` attribute with an integer value in the range of `"1"` to `"9"` on the `<sf:p>` element. An `sf:restart-list` attribute with a value of `"true"` indicates that the paragraph begins a new sequence at this level rather than continuing a previous one.

The appearance of the list element—that is, bullets, indentation, and so on—is found in the paragraph style.

Multiple Column Text Objects

"Figure 4-3" illustrates a Pages text object with a page layout comprised of multiple columns. In this case, the text flows and wraps over all three columns. "Listing 4-3" describes the XML for text in multiple columns.

Figure 3-3 Example output for a page layout with multiple columns



Listing 3-3 The XML for a multiple column page layout

```
<sf:text-storage sf:kind="body" sfa:ID="SFWPStorage-8">
<sf:stylesheet-ref sfa:IDREF="SFSStylesheet-1"/>
<sf:text-body>
  <sf:section sf:name="Chapter 2" sf:style="section-style-22">
    <sf:layout sf:style="layout-style-35">
      <sf:p sf:style="paragraph-style-32">Text in the second
section.<sf:layoutbr/></sf:p>
    </sf:layout>
    <sf:layout sf:style="SFWPLayoutStyle-13">
       <sf:p sf:style="paragraph-style-32">New layout  with multiple columns.
New  layout with multiple columns. New layout with multiple columns.  New layout
 with  multiple columns. New layout with multiple columns. New layout with
multiple  columns. New layout with multiple columns. <sf:layoutbr/>
      </sf:p>
    </sf:layout>
    <sf:layout sf:style="layout-style-36">
      <sf:p sf:style="paragraph-style-32">
          <sf:br/>
      </sf:p>
    </sf:layout>
  </sf:section>
</sf:text-body>
</sf:text-storage>
```

Defining Characteristics

The `<sf:layout>` element indicates the layout style for the multiple column text view that is generated in "Figure 4-3." Note that the column count is a property of the layout style.

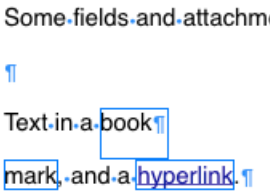
The `sf:style` attribute specifies the paragraph style, which includes the "New layout with multiple columns" text, followed by a `sf:layoutbr` break attribute.

The `<sf:layoutbr>` element is the most notable item of interest in this XML example. The last paragraph in a layout is terminated by an `<sf:layoutbr>` instead of a paragraph break.

Text Objects With Date-Time Fields and Hyperlinks

"Figure 4-4" illustrates the output for a Pages text object with a page layout comprised of a date-time field, along with page numbers, bookmarks, and hypertext links. "Listing 4-4" describes the XML for this text-storage object.

Figure 3-4 Example output for a text object with date-time fields, page numbers, bookmarks, and hypertext links



Listing 3-4 The XML for date-time fields, page numbers, bookmarks, and hypertext links

```
<sf:text-storage sf:kind="body" sfa:ID="SFWPStorage-8">
<sf:stylesheet-ref sfa:IDREF="SFSStylesheet-1"/>
<sf:text-body>
  <sf:section sf:name="Chapter 2" sf:style="section-style-22">
    <sf:layout sf:style="layout-style-36">
      <sf:p sf:style="paragraph-style-32">Some fields and attachments:
          <sf:date-time sf:format="MMMM d, yyyy h:mm a" sf:locale="en_US"
              sf:date-style="long" sf:time-style="short"
             sf:auto-update="true">April 19, 2005 11:46 PM</sf:date-time>  page
 number:
          <sf:page-number sf:value="2"/> of:
          <sf:page-count sf:value="2"/>.
          <sf:br/>
       </sf:p>
      <sf:p sf:style="paragraph-style-32">
        <sf:br/>
      </sf:p>
      <sf:p sf:style="paragraph-style-32">Text in a
          <sf:bookmark sf:name="bookmark" sf:ranged="true" sf:page="2"
sfa:ID="SFWPBookmarkField-0">
book<sf:br/></sf:bookmark>
      </sf:p>
      <sf:p sf:style="paragraph-style-32"><sf:bookmark-ref
sfa:IDREF="SFWPBookmarkField-0">mark
</sf:bookmark-ref>, and a <sf:link href="http://web.apple.com"><sf:span
sf:style="SFWPCharacterStyle-6">
hyperlink</sf:span></sf:link>.<sf:br/></sf:p>
    </sf:layout>
  </sf:section>
</sf:text-body>
</sf:text-storage>
```

Defining Characteristics

This section discusses the defining characteristics of the various fields of this text object, including bookmark, placeholder text, hyperlink, date-time, and page number and page count.

Bookmark Fields

A bookmark field is defined by a `<sf:bookmark>` element with no attributes other than an optional `sfa:ID`, which only needs to be defined if the bookmark crosses a paragraph boundary, in which case the continuation is indicated by a `<sf:bookmark-ref>` element with an `sfa:IDREF` attribute with a value corresponding to the value of the `sfa:ID` of the `<sf:bookmark>` element that is being continued.

Placeholder Text Fields

A placeholder text field is defined by a `<sf:ghost-text>` element with no attributes other than an optional `sfa:ID`, which only needs to be defined if the placeholder text crosses a paragraph boundary, in which case the continuation is indicated by a `<sf:ghost-text-ref>` element with an `sfa:IDREF` attribute with a value corresponding to the value of the `sfa:ID` of the `<sf:ghost-text>` element which is being continued. A placeholder text field should contain textual content only.

Hyperlink Fields

A hyperlink field is defined by a `<sf:link>` element with a single attribute. A hyperlink field must not cross a paragraph boundary, and cannot contain graphical attachments, date-time fields, placeholder text, or other hyperlink fields. It may contain `<sf:span>` elements.

The `href` attribute defines either a standard web (`http:`) or mail (`mailto:`) URL, or a bookmark name prefix by '#'.

Date-Time Fields

A date-time field is defined by a `<sf:date-time>` element with five possible attributes, and no child elements. The content of the field is text only, and is the resulting formatted text seen in the document.

The `sf:format` attribute defines the format of the date or time text. The syntax of this string is defined by the ICU library http://icu.sourceforge.net/userguide/formatDateTime.html. (Only a small selection of possible formats are exposed in the UI.)

The `sf:locale` attribute specifies the locale used to format the date time string. The possible values are those defined by CFLocale.

The optional `sf:date-style` and `sf:time-style` attributes define the style indicated by the format in the `sf:format` attribute. The possible values are `"short"`, `"medium"`, `"long"`, and `"full"`, which correspond to `CFDateFormatterStyle` values. These attributes are defined only if together they result in the format specifed by the value of the `sf:format` attribute; otherwise, they are omitted.

If the `sf:auto-update` attribute has a value of `"true"`, then the field is auto-updating.

Page Number and Page Count Fields

A page number field is defined by a `<sf:page-number>` element with a single attribute and no content.

A page count field is defined by a `<sf:page-count>` element with a single attribute and no content.

The value of the `sf:value` attribute of page number and page count fields is an integer corresponding to the most recent value represented by the field, and is recomputed whenever the text containing the field is laid out.

A Footnote Text Object

"Figure 4-5" illustrates the output for a Pages text object with two footnotes and their footnote references (the superscripted numbers in the body text). "Listing 4-5" describes the XML for this text-storage object.

Figure 3-5 Example output for a text object with footnotes

Listing 3-5 The XML for a footnotes text object

```
<sf:text-storage sf:kind="body" sfa:ID="SFWPStorage-8">
<sf:stylesheet-ref sfa:IDREF="SFSStylesheet-1"/>
<sf:footnotes>
  <sf:text-storage sfa:ID="SFWPStorage-7" sf:kind="footnote">
    <sf:stylesheet-ref sfa:IDREF="SFSStylesheet-1"/>
    <sf:attachments/>
    <sf:text-body>
      <sf:p sf:style="kSFWPFootnoteTextStyleIdentifier"><sf:span
sf:style="SFWPCharacterStyle-4">
<sf:footnote-mark sf:mark="1"/></sf:span> First footnote<sf:footnotebr/></sf:p>
      <sf:p sf:style="kSFWPFootnoteTextStyleIdentifier"><sf:span
sf:style="SFWPCharacterStyle-4">
<sf:footnote-mark sf:mark="2"/></sf:span> This is the footnote
text.<sf:footnotebr/></sf:p>
      <sf:p sf:style="paragraph-style-default"/>
    </sf:text-body>
  </sf:text-storage>
</sf:footnotes>
<sf:text-body>
  <sf:section sf:name="Chapter 2" sf:style="section-style-22">
    <sf:layout sf:style="layout-style-36">
      <sf:p sf:style="paragraph-style-32">Footnotes<sf:span
sf:style="SFWPCharacterStyle-4">
```

```
<sf:footnote sf:index="0" sf:autonumber="1"/></sf:span>, too<sf:span
sf:style="SFWPCharacterStyle-4">
<sf:footnote sf:index="0" sf:autonumber="2"/></sf:span><sf:br/></sf:p>
      <sf:p sf:style="paragraph-style-32">
        <sf:br/>
      </sf:p>
    </sf:layout>
  </sf:section>
</sf:text-body>
</sf:text-storage>
```

Defining Characteristics

This section discusses the defining characteristics of the various fields of this text object, including footnotes and footnote references.

Footnotes

The `<sf:footnotes>` element defines a single `<sf:text-storage>` element that must have a `sf:kind` value of `"footnote"`.

Within the footnote storage, individual footnotes are separated by `<sf:footnotebr>` elements.

The footnote numbers are indicated by optional `<sf:footnote-mark>` elements with a single attribute. The value of the `sf:mark` attribute defines the number given to the footnote.

Footnote References

The footnote references are defined in the body text as `<sf:footnote/>` elements with two attributes and no content.

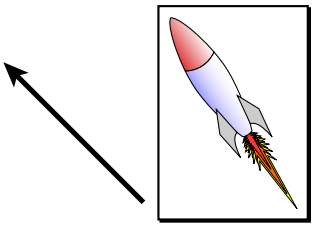
The value of the `sf:index` attribute defines a valid index into the text of the `<sf:footnotes>` element. (Zero for the first footnote and so on.)

If the value of the `sf:autonumber` attribute is `"YES"`, then the `sf:mark` attribute on the corresponding `<sf:footnote-mark>` is renumbered as footnotes are added and removed.

A Text Object With an Image and a Drawable Attachment

"Figure 4-6" illustrates a Pages text object with an image attachment and a drawable attachment. "Listing 4-6" describes the XML for this text-storage object.

Figure 3-6 Example output for a text object with an image and a drawable attachment



Listing 3-6 The XML for a text storage object with an image and a drawable attachment

```
<sf:text-storage sf:kind="body" sfa:ID="SFWPStorage-8">
  <sf:stylesheet-ref sfa:IDREF="SFSStylesheet-1"/>
  <sf:attachments>
    <sf:attachment sfa:ID="SLDrawableAttachment-0" sfa:sfclass=""
sf:kind="drawable-attachment">
      <sf:line sfa:ID="SFDLineInfo-0">
      </sf:line>
    </sf:attachment>
    <sf:attachment sfa:ID="SLDrawableAttachment-1" sfa:sfclass=""
sf:kind="drawable-attachment">
      <sf:image sfa:ID="SFDImageInfo-0">
      </sf:image>
    </sf:attachment>
  </sf:attachments>
  <sf:text-body>
    <sf:section sf:name="Chapter 2" sf:style="section-style-22">
    <sf:layout sf:style="layout-style-36">
      <sf:p sf:style="paragraph-style-32">
      <sf:attachment-ref sfa:IDREF="SLDrawableAttachment-0"
sf:kind="drawable-attachment"/>
      <sf:attachment-ref sfa:IDREF="SLDrawableAttachment-1"
sf:kind="drawable-attachment"/>
      </sf:p>
    </sf:layout>
    </sf:section>
  </sf:text-body>
</sf:text-storage>
```

Defining Characteristics

This section discusses the defining characteristics of the various fields of this text object, including attachments and attachment references.

Attachments

The `<sf:attachments>` element defines a sequence of `<sf:attachment>` elements, each of which defines a single attachment appearing in the following `<sf:text-body>` element.

The `<sf:attachment>` element has two attributes (in addition to an `sfa:ID attribute`) and has a child element that is either a `<sf:line>`, `<sf:drawable-shape>`, or `<sf:image>`.

The `sfa:class` attribute must always have a value of `""`.

The `sf:kind` attribute has a value of `"drawable-attachment"` if the child element is an `<sf:line>`, `<sf:drawable-shape>`, or `<sf:image>`.

Attachment References

Within the text body, attachment points are defined by `<sf:attachment-ref>` elements with one attribute, in addition to the `sfa:IDREF` attribute that references the attachment in the `<sf:attachments>` element.

The `sf:kind` attribute must have the same value as the `sf:kind` attribute on the referenced attachment.

Document Revision History

This table describes the changes to *iWork Programming Guide*.

Date	Notes
2005-11-09	Revised section introducing variations in Keynote shapes. Minor editorial fixes throughout. Reformatted XML code samples.
2005-09-08	First public release of document for iWork developers.

Document Revision History