Mail Programming Topics

Apple Applications > Mail

2007-05-22

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Figures and Tables

Introduction

This document describes how to customize some of the features in Mail. Even though Mail does not contain an API, some features can be tailored using property lists and HTML. The articles that follow explain how to format these property list files and where to install them.

Note: This document is a series of related articles intended to be read in any order. The articles are being published in serial fashion—new articles will be added periodically.

This document is intended for developers who want to add to the functionality of Mail, or for system administrators who want to control the user's setup process for Mail.

Organization of This Document

This document contains the following articles:

- "Creating Mail Account Bundles" (page 9) explains how to create an bundle to set up mail accounts.
- "Creating Mail Stationery Bundles" (page 15) explains how to create a stationery template in Mail.

Creating Mail Account Bundles

Starting with Mac OS X version 10.4, Mail has a new user interface for configuring accounts. The interface is a series of panels that prompts the user for information and verifies the information during the configuration process. You can provide users with a mail account bundle that streamlines the account setup process. Information such as the server name and connection options can be preset. Additionally, you can specify whether or not the user is allowed to change the values you have specified. There is also a setting that allows you to direct the user to a URL where they can get information to help with troubleshooting problems on their own. In this article, you will learn how to create a mail account bundle and you will see an example that is an excellent starting point for creating your own mail account bundles.

Creating a Bundle Directory

To create a mail account bundle, first create a new directory that will hold the files associated with the mail account bundle. To avoid name conflicts, the name of the mail account bundle directory should be the reverse-ordered ICANN domain name (for example, `com.apple`) and have the suffix `mailaccounts`. The bundle used to assist .mac users setting up mail accounts is installed in `/Library/Mail/AccountTypes` and is named `com.mac.mailaccounts`.

Next, create a XML file inside the bundle. It must be named `MailAccounts.plist`. This file should be an XML property list file with the keys listed in Table 1 (page 9). All keys are required unless explicitly indicated as optional. See the example at the end of this document for the proper nesting of keys.

Note: Some of the dictionary keys have been deprecated in Mac OS X v10.5. These keys may still be used for mail account bundles in Mac OS X v10.4 but are ignored in Mac OS X v10.5.

Table 1 Keys in `MailAccounts.plist`

Type	Description	Key
`AccountIconName` (deprecated)	String	Optional. The name of an icon file to use for this account. The icon should be 16 by 16 pixels and the file should be placed in the top level of the bundle directory.
`Accounts`	Array	An array of dictionary elements describing the account types. See "Account Dictionaries" (page 10) for details on the format of these dictionaries.
`Identifier`	String	The reverse-ordered ICANN domain name; for example, `com.mac`. This name must be unique.
`Version`	Integer	Currently, the `version` value must be `1`.

Account Dictionaries

Each dictionary in the Accounts array describes an account type that can be configured by the user. The keys in Table 2 (page 10)are supported. Unless indicated as optional, all keys are required for each account dictionary defined in the property list.

Table 2 Keys in each dictionary in `Accounts`

Key	Type	Description
`AccountID`	String	A CFUUID string, generated by running `uuidgen` in Terminal. For more information about CFUUIDs, see *CFUUID Reference*.
`AccountName`	String	The default account name for an account.
`AuthenticationScheme`	Array	Optional. An array of supported authentication schemes. An empty array or missing key causes accounts to be configured using `Password` security. List all authentication schemes that are supported by the account. The first item in the list is the preferred scheme. Supported schemes are: • `Password` • `CRAM-MD5` • `GSSAPI` • `KERBEROS_V4` (deprecated) • `X_KPOP` (deprecated) Both `KERBEROS_V4` and `X_KPOP` are now treated as `GSSAPI` in Mac OS X v10.5.
`AuthenticationScheme-IsEditable` (deprecated)	Boolean	Optional. Indicates whether the user is allowed to change the authentication scheme.
`DeliveryAccounts`	Array	Optional. This is an array of CFUUIDs associated with mail delivery accounts defined in the same mail account bundle. The first outgoing account defined in the array is used by default.
`EmailAddressDomain`	String	Optional. Indicates the domain name associated with the e-mail address. Mail uses this key to match the user's email address to the enclosing account dictionary.
`PortNumber`	Integer	Optional. A non-standard port number to use when connecting to the server. When specified, Mail attempts to connect to the port first before trying standard port numbers.
`PortNumberIsEditable` (deprecated)	Boolean	Optional. Indicates whether the user is allowed to change the port number used when connecting to the server.

Key	Type	Description
`Protocol`	String	Indicates what protocol to use when communicating with the server. Supported values are: • `IMAP` • `POP` • `SMTP`
`SSLEnabled`	Boolean	Optional. Indicates whether SSL should be used when communicating with the server. If this key is not specified, Mail attempts an SSL connection to the server first and falls back to non-SSL if that fails.
`SSLEnabledIsEditable` (deprecated)	Boolean	Optional. Indicates whether the user is allowed to change the setting of whether SSL is used or not. If this key is set to true and Mail fails to connect to the server using SSL, the user is shown the Incoming Security Settings pane of the setup assistant.
`ServerName`	Array	A list of server names to allow the user to pick from. The first server name is the preferred server and is used by default.
`ServerNameIsEditable` (deprecated)	Boolean	Optional. Indicates whether the user is allowed to change the server name.
`SupportURL`	String	Optional. The URL shown in the Mail Account Preferences panel that takes the user to the specified webpage for more help or information.
`SupportURLLabel`	String	Optional. When present, the specified value is displayed instead of the `SupportURL` value in the Mail account preference panel. Clicking on the label opens the URL stored in `SupportURL`.
`UserNameIsEmailAddress`	Boolean	Optional. Indicates whether the user name is the user's full email address.
`UserNameMatchesEmailAddress`	Boolean	Optional. Indicates whether the user name and the portion of the user's email address to the left of the @ sign are the same value.

Localizing Mail Accounts

You can localize values for the `AccountName` and `SupportURL` keys in mail account bundles. This lets each account have its own localizations. For more information on localization, see the Internationalization Documentation.

Using Simplified Account Setup

Mail provides a new feature in Mac OS X v10.5 that allows users to add a new account with only their full name, email address, and password. This feature is known as simplified account setup.

Mail account bundles are automatically used for simplified account setup if the following conditions are met:

- The `EmailAddressDomain` dictionary key must be specified.
- The `UserNameIsEmailAddress` or `UserNameMatchesEmailAddress` dictionary key must be set to true.
- The `DeliveryAccounts` dictionary key must have exactly one CFUUID.

Installation

To install a mail account bundle, you need to place the top-level directory, which contains the `MailAccount.plist` file, in one of the following locations:

- `~/Library/Mail/AccountTypes`
- `/Library/Mail/AccountTypes`
- `/Network/Library/Mail/AccountTypes`

A Mail Bundle Property List Example

The following example is the property list used to create the .Mac mail account bundle shipping with Mac OS X v10.4 and later:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Accounts</key>
    <array>
        <dict>
            <key>AccountID</key>
            <string>5D795DD4-55CB-11D8-A81E-000A957054BE</string>
            <key>AccountName</key>
            <string>.Mac</string>
            <key>AuthenticationScheme</key>
            <array>
            <array/>
            <key>DeliveryAccounts</key>
            <array>
                <string>5C5150FA-55CB-11D8-A746-000A957054BE</string>
            </array>
            <key>EmailAddressDomain</key>
            <string>@mac.com</string>
            <key>Protocol</key>
            <string>IMAP</string>
            <key>ServerName</key>
            <array>
                <string>mail.mac.com</string>
            </array>
            <key>UserNameMatchesEmailAddress</key>
            <true/>
        </dict>
        <dict>
            <key>AccountID</key>
            <string>5C5150FA-55CB-11D8-A746-000A957054BE</string>
            <key>AccountName</key>
            <string>.Mac SMTP</string>
            <key>AuthenticationScheme</key>
            <array>
                <string></string>
            </array>
            <key>Protocol</key>
            <string>SMTP</string>
            <key>ServerName</key>
            <string>smtp.mac.com</string>
        </dict>
    </array>
    <key>Identifier</key>
    <string>com.mac</string>
    <key>Version</key>
    <integer>1</integer>
</dict>
</plist>
```

Creating Mail Stationery Bundles

Mail Stationery is a new feature in Mac OS v10.5 that allows users to customize their email messages with predefined stationery templates. Each stationery template is a bundle that contains all the files needed to display the stationery.

Stationery Bundles

A stationery template is a bundle folder with the file extension `.mailstationery`. Inside the root folder is a Contents folder, and inside that is a Resources folder. All the files in the stationery bundle are located in the Resources folder.

Description Property List

The `Description.plist` file serves as a table of contents describing all the other files in the stationery folder. The Description is a dictionary with the following keys and values:

- Folder Name—The name of the stationery bundle folder on disk, including the `.mailstationery` extension. Required.
- Display Name—The name of the template as it is shown to the user. This entry isn't necessary if a display name is given in the category's table of contents or if a localized display name is available. If no display name is supplied in any of these locations, the folder name, minus the `.mailstationery` extension, is used as the display name. For information about localizing this string, see "Localization" (page 21).
- Stationery ID—The unique identifier for the stationery template. If this entry is missing, Mail generates one.
- HTML File—The name of the HTML file, including any extensions. Required.
- Thumbnail Image—The name of the file containing the thumbnail image of the template. The image is shown in the stationery pane of a New Message window. Required.
- Background Images—An array of the names of any images referenced as background images in the HTML. Required if the template has any background images.
- Tiled Background Image—The name of the image that serves as a background for the entire template. This name should also appear in the Background Images entry. Required if the template has a background image for the body element or another element that encompasses the entire visual content.
- Images—An array of the names of any local images referenced in `IMG` elements in the HTML. Required if the template's HTML has any `IMG` elements with local URLs.

- Composite Images—An array of dictionaries, each dictionary representing one composite image referred to in the template's HTML file. The keys and values of a composite image's dictionary are described below. See "Composite Images" (page 17) for more information about the `IMG` tag and the component images. Each composite image referred to in the HTML file must have an entry. Each of the keys described below is required if the image it identifies is present.
    - ❏ Composite Name—The value of the `SRC` attribute in the `IMG` tag that represents the composite image.
    - ❏ Base Image—The filename of the image that forms the base, or background, of the composite image.
    - ❏ Overlay Image—The filename of the image to be used as the top layer of the composite image.
    - ❏ Masks—An array of dictionaries, with each dictionary corresponding to one mask file. The dictionaries should be in left-to-right, top-to-bottom order of the drop zones they describe. Each dictionary should have the following keys. For the values that involve a coordinate system, the origin should be assumed to be at the bottom-left corner of the mask image, and the units are in pixels.
        - ❏ File Name—The filename of the mask image.
        - ❏ Placeholder Image—The filename of the placeholder image.
        - ❏ Drop Zone Offset—The position where the bottom-left corner of the user's picture should be placed. This is an array of integers (x, y).
        - ❏ Drop Zone Angle—The angle, measured in degrees, through which the user's picture should be rotated. Angles are measured counterclockwise.
        - ❏ Drop Zone Size—The width and height to which the user's picture should be scaled. This should be an array of integers (width, height).

HTML File

The HTML file defines the content of the stationery. It can contain any valid HTML that is displayed correctly in common email clients. It should not contain any JavaScript. The next few sections describe the other specific types of content and HTML elements.

HTML Content

By default, the content of the stationery is not be editable by the user. Each region of the document that the user can edit must be surrounded by an element (for example, `DIV` or a `SPAN`) with the `contenteditable=YES` attribute.

Precautions must be taken to ensure compatibility with common email client applications, which are often much less capable than most web browsers at displaying HTML.

Dynamic Elements

Dynamic elements are placeholders that can be inserted in the HTML of the stationery. When the stationery is loaded into the New Message window, the dynamic element is replaced by static content. A dynamic element is specified in the HTML by a `SPAN` element with the class name `AppleStationeryDynamicElement`. Any HTML contained in the `SPAN` element is discarded when the stationery is loaded except when desired content cannot be inserted. On these occasions, the contents of the `SPAN` is displayed. The span tag must also have a `apple-content-type` attribute that specifies what the replacement content should be. For example, a value of `full-name` for `apple-content-type` specifies that the dynamic element should be replaced by the user's full name. The following values for `apple-content-type` are available:

`full-name`	User's full name from the Me card in Address Book.
`first-name`	User's first name from the Me card in Address Book.
`email`	The email address of the account being used to send the message.
`date`	The current date.

A dynamic element of type `date` must also have a `date-format` attribute. Its value is a format string as used by Cocoa's `NSCalendarDate` class.

Images

Local images that are supplied as resources in the stationery bundle and referenced in `IMG` tags or as background images are embedded in the message when it's sent. The URL in each `IMG` tag or background attribute should be a path relative to the stationery bundle's Resources folder. All `IMG` elements should have width and height attributes with absolute, not relative, values.

Composite Images in HTML

A composite image consists of one or more images from the stationery bundle that are layered together with pictures supplied by the user to create a single image.

A composite image is specified in the HTML with an `IMG` tag whose `SRC` attribute is the name of the composite image. This name is the `Composite Name` specified in the Description file. The end user may see this name, so it should be something that would be appropriate as a filename. The `IMG` tag should have the class name `AppleCompositeImage`.

For more information about composite images, see "Composite Images" (page 17).

Images

Images referenced by `IMG` tags or as background images should be included in the Resources folder of the stationery bundle. They should also be listed in the `Images` or `Background Images` entry in the Description. When the message is sent, the data for these images is included in the message.

Composite Images

Composite images provide a way for templates to include images supplied by the user in more attractive and complex ways than ordinary messages can supply. A composite image provides a static background image containing specific regions where the user can drop his or her own pictures. The user's pictures is sized and rotated as needed to fit the drop zones. After the user pictures is layered on top of the base or background image, an overlay image, usually mostly transparent, may be composited on top if desired.

A composite image is made up of a background image, one or more masks that define the drop zones for the user's pictures, and optionally an overlay image image to be composited on top of the user pictures. It's also possible to provide placeholder user pictures that are shown in the drop zones until the user replaces them with his or her own pictures.

Each mask image defines a region where the user can drop a picture. The only data in the mask that is used is the alpha channel. When the user drops an image in the drop zone, the image is be rotated to match the bottom edge of the drop zone if it isn't horizontal. It is then sized, maintaining its proportions, so that it just covers the entire drop zone, and positioned over the drop zone. Next, the alpha from the mask is set as the alpha for the user's image.

When the message is sent, the components of the composite image and the user pictures are composited together to form a single image, which is sent as part of the message. The individual components and the individual user pictures are not sent.

If the stationery's HTML contains any references to composite images, the files used to produce the composite image (except for the ones that are supplied by the user, of course) should be included in the Resources folder of the stationery bundle. They should be identified with the Base Image, Overlay Image, and Masks keys in the Description section, as described above.

The final image is created with the Generic RGB colorsync profile so any other images in the tempate whose colors need to match this one should have that same colorsync profile.

Thumbnail Image

Each stationery bundle should contain a thumbnail image of the stationery. The thumbnail image should have dimensions of 69x90 pixels.

Color-Changing Stationery

Some stationery templates can change colors when the user clicks the background in a New Message window or clicks on the thumbnail image in the stationery selection pane. For example, see Apple's Daisies template. For such a template, the HTML file is almost the same for an ordinary template, but the bundle contains multiple versions of some or all of the images and the Description dictionary contains some additional and modified entries to describe the color changes. Color changes can affect background images, foreground images, and text fonts that are specified in font elements in the HTML.

Image Files

For any image that changes color, the stationery bundle should contain a separate image file for each of the desired colors. An image that doesn't change color when the template changes needs only one image file. The changes needed in the Description file are described below. The names of the colors used in these entries are not visible to the user.

Description Property List

- Images—For a template that can change colors, each of the image names in this array should be replaced by a dictionary whose keys are the available colors. The value for each color should be the name of an image file. Even if an image doesn't change when the stationery changes colors, its entry must still be a dictionary. The dictionary of every entry must contain the complete set of colors.
- Background Images—This entry should be modified as described above for the Images entry.
- Composite Images—Base Image and Overlay Image. These entries should be modified as described above for the Images entry.
- Font Colors—This entry provides a means for one or more fonts specified in the HTML to change color. The entry is a dictionary whose keys are class names of HTML font elements whose colors need to change. See the section describing the HTML file for more information about the class names. The value associated with each class name is a dictionary whose keys are the available colors and whose values are the HTML color values. These color values should be the strings that would appear in the HTML, for example `#a4b936`.
- Default Color—This entry gives the name of the color that loads when the user first chooses the template. Changing this entry does not cause a different color to load initially; rather, this entry tells the application which set of images and font colors are referenced in the HTML.
- Letter Rectangle—This entry describes the portion of the stationery outside of which the user can click to change the color. Normally, this region is a rectangle that looks like a piece of paper. It's a dictionary with two keys–Top Margin and Width. The value of Top Margin should give the distance in pixels between the top of the view and the top of the letter region. Width should give the width in pixels of the letter region. Mail assumes that the HTML is written so that the letter region is always centered in the view.

HTML File

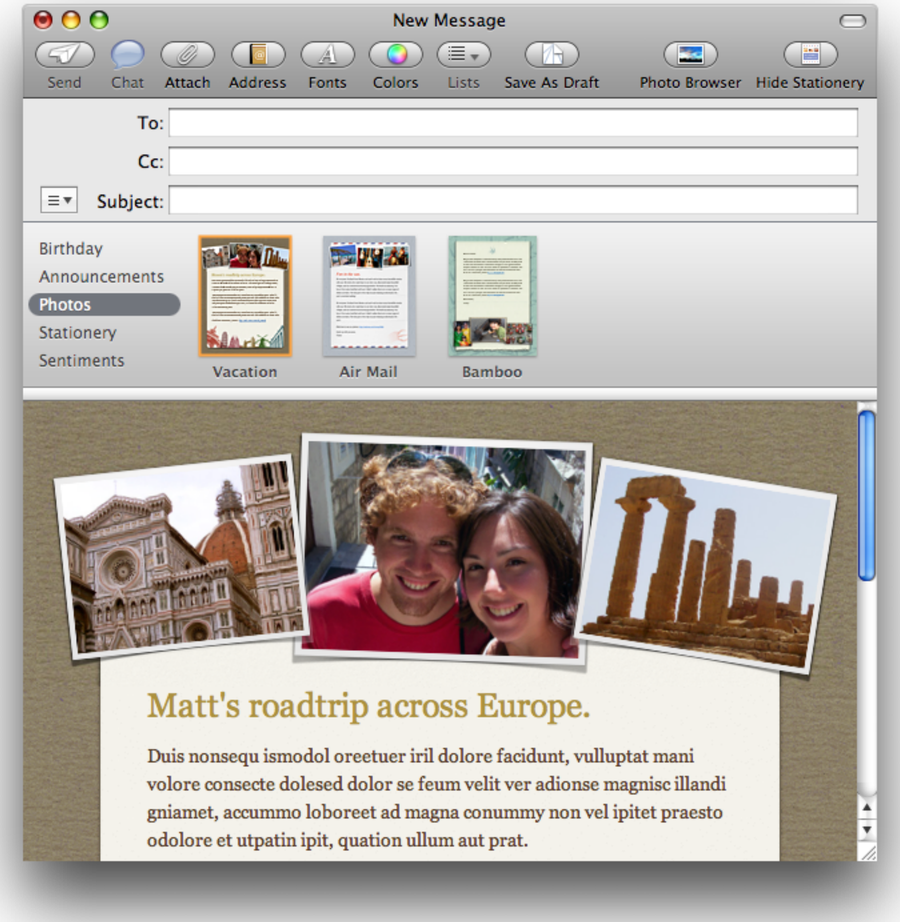
In order for a font color to change, it must be specified as a color attribute in a font element in the HTML, and the font element must have a class name. All the font elements that have the same color values can share the same class name. This class name is used as a key in the Font Colors entry of the Description file.

Stationery Installation

Stationery bundles are installed in one of two locations: `/Library/Application Support/Apple/Mail/Stationery` or `~/Library/Application Support/Mail/Stationery`. However, stationery bundles are not placed directly in these folders. At the root level of either of these locations is a company folder. By default, there is already an Apple company folder for all the preinstalled stationery templates. This company folder is referred to as a **company bundle**.

Like the stationery bundle, the company bundle contains a table of contents property list and a Contents folder which, in turn, contains a Resources folder. The Resources folder can contain a table of contents property list along with a folder for each category of stationery. When the user runs Mail, the stationery templates from all the company bundles are merged together when presented to the user. If multiple category bundles are found with the same display name, their templates are merged into a single list and the user sees them as a single category. See Figure 1 (page 20).

Figure 1 Mail stationery categories



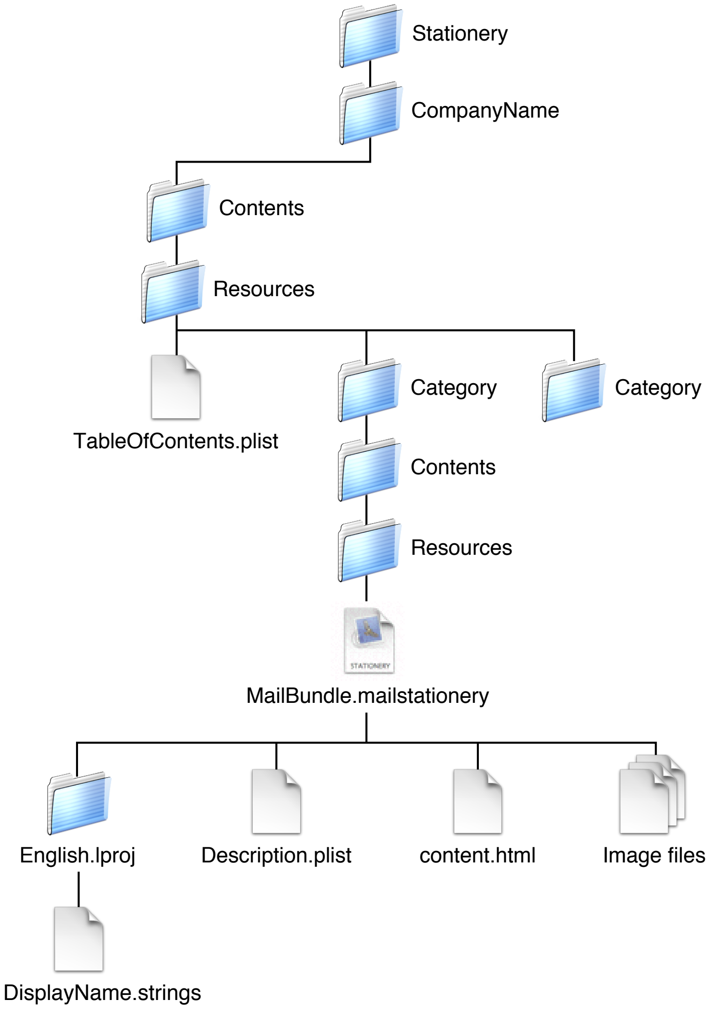
Matt's roadtrip across Europe.

Duis nonsequ ismodol oreetuer iril dolore facidunt, vulluptat mani volore consecte dolesed dolor se feum velit ver adionse magnisc illandi gniamet, accummo loboreet ad magna conummy non vel ipitet praesto odolore et utpatin ipit, quation ullum aut prat.

The table of contents file contains an an array of dictionaries, each of which describes one category (for a company bundle) or one stationery bundle (for a category bundle). The order of the entries in the dictionary determines the order in which the stationery templates or categories are displayed to the user. If the table of contents is missing or incomplete, Mail finds the stationery bundles or categories missing from the table of contents and displays them to the user.

The mail stationery bundles are installed within the category folders, known as **category bundles**. If the stationery is properly installed, the directory structure should look like Figure 2 (page 21).

Figure 2 Stationery directory structure



Localization

Some of the resources in the stationery, category, and company bundles can be localized using localization project folders. Mail uses the values in the `English.lproj` folder as defaults when it can't find values localized for other languages.

Stationery Bundles

You can localize the stationery bundle's display name by putting it in a localized strings file in the stationery bundle instead of in the description property list or the category's table of contents. The strings file should be named `DisplayName.strings`, and the key for the entry should be `Display Name`.

Category Bundles

The display name of a category can be localized by putting it in a localized strings file in the category bundle instead of in the `TableOfContents.plist` file of the company bundle. The strings file should be named `DisplayName.strings`, and the key for the entry should be `Display Name`.

Document Revision History

This table describes the changes to *Mail Programming Topics*.

Date	Notes
2007-05-22	New document that explains how to customize Mail features.