

MailMaestro™ Administrator's Guide

Copyright ©2002 Fastraq Limited. All rights reserved.
Revision 0.9.3

Table of Contents

| | | |
|------|------------------------------------|----|
| 1. | Introduction | 4 |
| 1.1. | An advanced Mail Server | 4 |
| 1.2. | Typical MailMaestro Systems | 5 |
| 1.3. | How MailMaestro Works..... | 6 |
| 2. | MailMaestro Operation..... | 7 |
| 2.1. | MailMaestro Users | 7 |
| 2.2. | MailMaestro Administrators..... | 9 |
| 2.3. | The MailMaestro Web Site..... | 11 |
| 3. | Installing MailMaestro | 14 |
| 3.1. | Requirements | 14 |
| 3.2. | Running the Installation | 15 |
| 4. | Configuring MailMaestro | 18 |
| 4.1. | The Administration Interface | 18 |
| 4.2. | Global Configuration | 18 |
| 4.3. | Region Configuration..... | 20 |
| 4.4. | Pricing Options | 21 |
| 4.5. | Billing Configuration..... | 23 |
| 5. | MailMaestro Implementation | 25 |
| 5.1. | The Mail Server | 25 |
| 5.2. | Administrative Interface | 27 |
| 6. | General Configuration | 29 |

- 6.1. Global Configuration 29
- 7. Account Configuration 31
 - 7.1. Accounts Directory Files..... 31
- 8. MailMaestro Web Site API..... 38
 - 8.1. Types 38
 - 8.2. Support Functions..... 39
 - 8.3. Configuration Files..... 42
 - 8.4. Log Functions..... 44
 - 8.5. Account Management Functions 45
 - 8.6. Message Access 47
 - 8.7. Domain Management 48
 - 8.8. Sending Messages..... 49
 - 8.9. Price Lists 50
 - 8.10. Billing System..... 52
 - 8.11. Billing Kits..... 54
 - 8.12. Bandwidth Charging 55
 - 8.13. Regions..... 57
 - 8.14. Context Sensitive Help..... 58

1. Introduction

Brought to you by the developers of Mailtraq and Mailkeep, MailMaestro will change the way you provide Internet services. MailMaestro is a turnkey mail server solution that gives your users not only the most flexible email service available, but also complete control over their own email management.

MailMaestro is a Mail Server designed to manage e-mail receipt for multiple domains. Specifically these domains can be managed independently of each other. This is an important distinction because normally a mail server administrator will be able to access and manage all the settings in the system. By only allowing domain administrators to manage their own domain, it becomes possible for one system to host the mail services for many independent users.

1.1. An advanced Mail Server

MailMaestro provides very flexible client-side access to mail. Most mail servers are only capable of operating in two modes :—

- store and forward (to other mail servers) or
- store for POP3 collection

It is generally recognised, though, that POP3 is only suitable for single users. That means that a (multi-user) company wanting its mail managed externally (i.e. without requiring a permanent, secure and reliable Internet presence) must employ complex and unreliable mail routing techniques.

MailMaestro, however, provides mail access using dynamic store-and-forward (which supports dynamic IP allocation for temporary/dialup customers) and store-for-collection using not only POP3 (and secure APOP) but also ETRN and ODMR. In fact MailMaestro was the first public mail service to support ODMR.

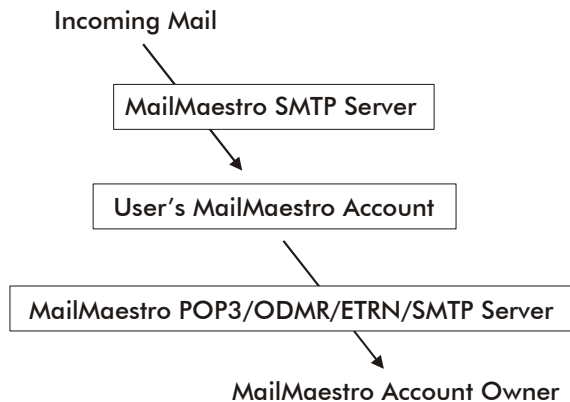


Figure 1 — Route mail takes to reach user account

1.2. Typical MailMaestro Systems

MailMaestro can be readily implemented as a chargeable Internet service, where accounts can be sold to users. Each account might hold one or more domains, and the account administrator would only have access to those domains.

Thus MailMaestro lends itself to a number of environments :—

- Internet Service Providers who have a customer base that needs the advanced mail services can offer MailMaestro accounts at a minimal cost.
- Consultants who manage and install mail systems for their clients can benefit by providing their clients with more comprehensive and reliable services with considerably less investment than providing complete Internet services.
- IT Administrators with requirements for high-availability, roaming users, distributed offices or high user turnover will be able to take advantage of MailMaestro's simple self-management interface.

Service Providers who already provide web services will find it easy to accommodate a MailMaestro system, as e-mail bandwidth requirements are much lower than other types of Internet service.

1.3. How MailMaestro Works

MailMaestro is a three-tier mail system. The **bottom tier** consists of a set of high-performance TCP/IP services optimized for reliability and heavy traffic. The system scales up and out effectively to handle large numbers of simultaneous users.

The **middle tier** is a JavaScript API that runs on Microsoft IIS. This tier provides centralized and standardised access to the control files used by the bottom tier. The API makes it possible to completely customize your web site and the web-based interfaces to MailMaestro with as little effort as possible and without sacrificing the proven reliability of the system. Microsoft IIS is only used for the web-based interface; it is not involved in the actual mail processing.

The **top tier** is a fully functional web site designed to make it as easy as possible for users to configure and manage their accounts, and for administrators to manage the MailMaestro system. Everything in the web site can be re-designed to your organization's style and objectives.

The source for the middle and top tiers is provided for maximum flexibility. There is nothing you cannot customize.

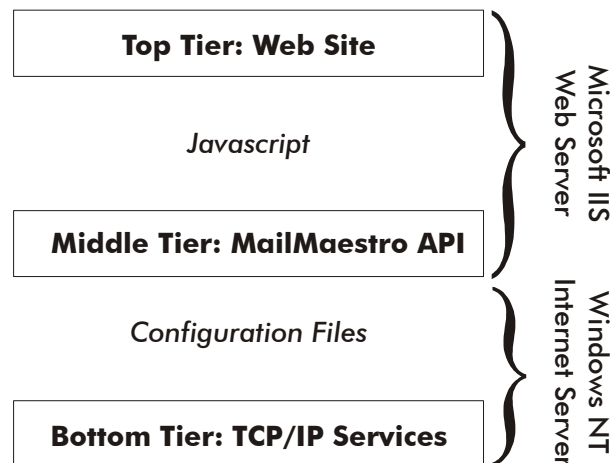


Figure 2 — MailMaestro's multi-tier structure

2. MailMaestro Operation

This section shows how a typical MailMaestro system operates on a day-to-day basis.

2.1. MailMaestro Users

The MailMaestro system is designed to allow users configure and maintain their own accounts without administrative involvement. MailMaestro also provides for on-line billing making it possible to provide a completely automated service.

2.1.1. Signing Up

Users can sign up easily by filling out a short form on the web site. This form includes their e-mail address (required for further communication), their location (required for regional pricing) and their name (to simplify communication). No other information is required at this stage.

To verify the user's e-mail address their account details (including access password) are e-mailed to them after they have completed the form.

Users are issued a unique account number. An account number is preferable as it is un-ambiguous. A domain is not used to identify an account as an account may have more than one domain, and those domains may change over time.

2.1.2. Trial Accounts

The user's account is initially in **trial** mode. Typically it will expire 14 days after creation, giving the user enough time to test the functionality. Trial mode is active until the user upgrades their account (which involves purchasing an appropriate pricing option).

***Note:** When in trial mode, users cannot delete domains from their account. This is important as it prevents users from continuously creating trial accounts to manage their mail for free.*

Accounts cannot be deleted manually. Instead, they are erased some period (typically 30 days) after expiry.

When an account has expired, it will not accept incoming mail. However, it will allow web access and allow mail collection (otherwise it may appear that the user's mail is being held from them).

2.1.3. Account Limitations

A MailMaestro account has a number of limitations, many dependent on the upgrade option that the user selects. (The MailMaestro administrator can define the available upgrade options.)

| | |
|------------------|---|
| Expiry | This is how long the account will last |
| Quota | This is how much mail can be stored in the account at any time |
| Bandwidth | This is how much mail can be sent to the account in any month |
| Domains | This is how many domains may be assigned to the account at any time |

Users can upgrade their account at any time. The new settings come into effect immediately, but the new expiry is determined by comparing the new settings to the current settings.

2.1.4. Excess Bandwidth

If an account has excess bandwidth enabled, the bandwidth limits become a free tier, after which the user must pay for the bandwidth they use. This charge is applied at the end of the month and the user must log in during the next month to pay for it, or the account is locked (as though the user exceeded their bandwidth in a non-excess setting).

2.1.5. User Account Management

Users can log on to their account via the web site and from there they can perform a number of tasks :—

- Add and remove domains

- Change access methods and their password
- View and delete messages in their account
- View logs of activity in their account
- Upgrade the account

2.1.6. Billing

Users can upgrade their account at any time. (Renewing their account is essentially the same as upgrading to the same tier, with the side effect of extending their expiry.)

When the user purchases an upgrade, they can pay through a billing kit determined by their region (thus imposing the appropriate taxes). An invoice is generated and this is stored in their account for future reference.

At the end of the month if the user's account has excess bandwidth enabled, and the account has exceeded the bandwidth limit, then a charge for the excess bandwidth is placed in the account. The user must log in and pay for this excess during the next month and that is billed and invoiced in the same way as an upgrade.

2.1.7. Typical Activity

Once the user has configured their account, they will rarely need to visit the web site (except to investigate problems or upgrade their account, or to pay for excess bandwidth usage).

All day-to-day usage takes place through the mail protocols.

2.2. MailMaestro Administrators

The administrator's role is to keep the system running smoothly, and provide support for users where necessary. One of MailMaestro's objectives is to minimise this workload by giving the user as much flexibility and control as possible, along with features such as log access to help them solve problems themselves.

The MailMaestro Administrative web site allows complete control over user accounts, as well as allowing the administrator to “log in” to a user account and see what they see.

Note: this “log in as” feature is limited in that the administrator cannot view the bodies of the messages in the account.

Administrators are not *required* to perform any routine maintenance on MailMaestro as the system can operate effectively without intervention. Administration is only required when faults occur, or when users have problems that they cannot resolve by themselves.

The administrators can view all the accounts and domains in the system and can modify any property in the accounts themselves. As a result, administrators can change quotas, extend accounts, increase bandwidth, etc without having to purchase a new pricing option.

Administrators will need to configure the billing aspect of the service, and the following sections describe this in more detail.

2.2.1. Regions

MailMaestro can allocate each account to a region (chosen when a user signs up for an account). Pricing options (described next) can be made visible only to specific regions (to allow for discounts, different currencies) and each region can have a different tax rate.

2.2.2. Pricing Options

There are two types of pricing options :—

- 1) Upgrade Pricing sets a price, bandwidth limit, domain limit, quota and usage term (time before expiry). A user can upgrade to this option at any time. The only variable is the usage term, which is extended if based on the time left on their current account (and the different in limitations, explained *****). Each upgrade pricing option can also, optionally, activate an excess charge option.
- 2) Excess Charges define the price-for-bandwidth applied to bandwidth used in a month over the account's limit.

2.2.3. Billing Kits

Billing Kits are methods for charging a user for an upgrade. Typically a billing kit interfaces with a payment provider (such as a credit-card gateway). MailMaestro comes with a sample kit, although it is relatively easy to create new kits to support both additional gateways and additional payment methods (such as purchase orders paid by cheque).

By keeping the billing kit concept separate, it is possible to alter the kits without interfering with the pricing options or other services.

2.3. The MailMaestro Web Site

The web site (top tier) consists of a user configuration section and an administration section. The user section is authenticated using the MailMaestro API using the account name and password.

The screenshot shows the MailMaestro web interface in Microsoft Internet Explorer. The browser title is "MailMaestro -- The SMTP and ODMR Mail Server - Microsoft Internet Explorer". The page has a navigation bar with links: home, manage your account, about mailmaestro, signup, help. The main header features the MailMaestro logo and the tagline "the turnkey internet mail service", along with a "return to admin" link.

The main content area is titled "Manage Your Account" for user "Elric Pedder (fastraq)". It displays the following information:

- Account Control** (left sidebar):
 - Account overview
 - Configure your domains
 - Configure mail collection options
 - Change your password
 - Contact e-mail addresses
 - View access logs
 - View messages in your account
 - Upgrade your account
 - View billing history
- Message Status:** You have 1 message(s) waiting in your account. [View your mail messages](#)
- Domain Status:** You have 1 domains assigned to your account. [Manage your domains](#)
- Collection Methods:** You may enable a variety of collection methods, including SMTP, ETRN, ODMR and POP3. [Configure mail collection](#)
- Log Files:** If you are encountering problems, or wish to monitor recent activity, you can view the log files recorded during previous mail collection sessions. [View log files](#)
- Access Management:**
 - [Change your access password](#)
 - Your external contact e-mail address is **elric@fastraq.co.uk**
 - Your administrative e-mail address is **elric@fastraq.co.uk**
 - [Change your e-mail addresses](#)
- Account Quotas:**

| | |
|-------------------|------------------|
| Data Limit: | 10mb |
| Data Used: | 101kb |
| Bandwidth Limit: | 4kb/month |
| Bandwidth Used: | 5kb |
| Excess Bandwidth: | £15.00 GBP / 4kb |
- Access Details:**

| | |
|----------------|---------------------------------|
| Last Accessed: | Thu, 11 Apr 2002 10:48:14 +0100 |
| From: | dns.novitraq.com |
| IP Address: | 204.92.85.2 |
| Access Type: | SMTP |
- Account Details:**

| | |
|------------------|------------|
| Account Expires: | 05/05/2002 |
|------------------|------------|

[Renew or Upgrade Account](#)

Figure 3 — User Account Management

The admin section exists in the admin sub-directory and relies on the Windows NT/2000 Server Security. Anonymous access should not be granted to this directory. When someone attempts to browse to this directory, IIS will request authentication from them (as shown in Figure 4).

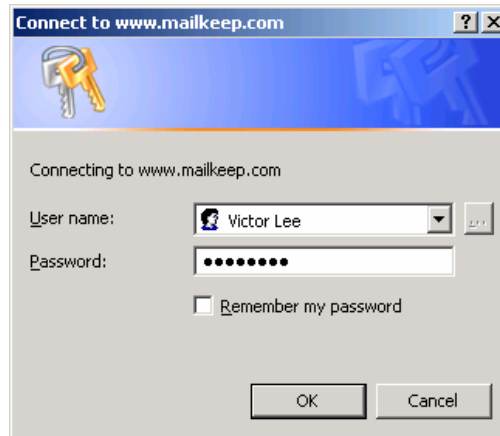


Figure 4 — Logging in to the Admin section

The administration section provides a means to monitor the MailMaestro system and manage user accounts. An example from the administration section is given in Figure 5.

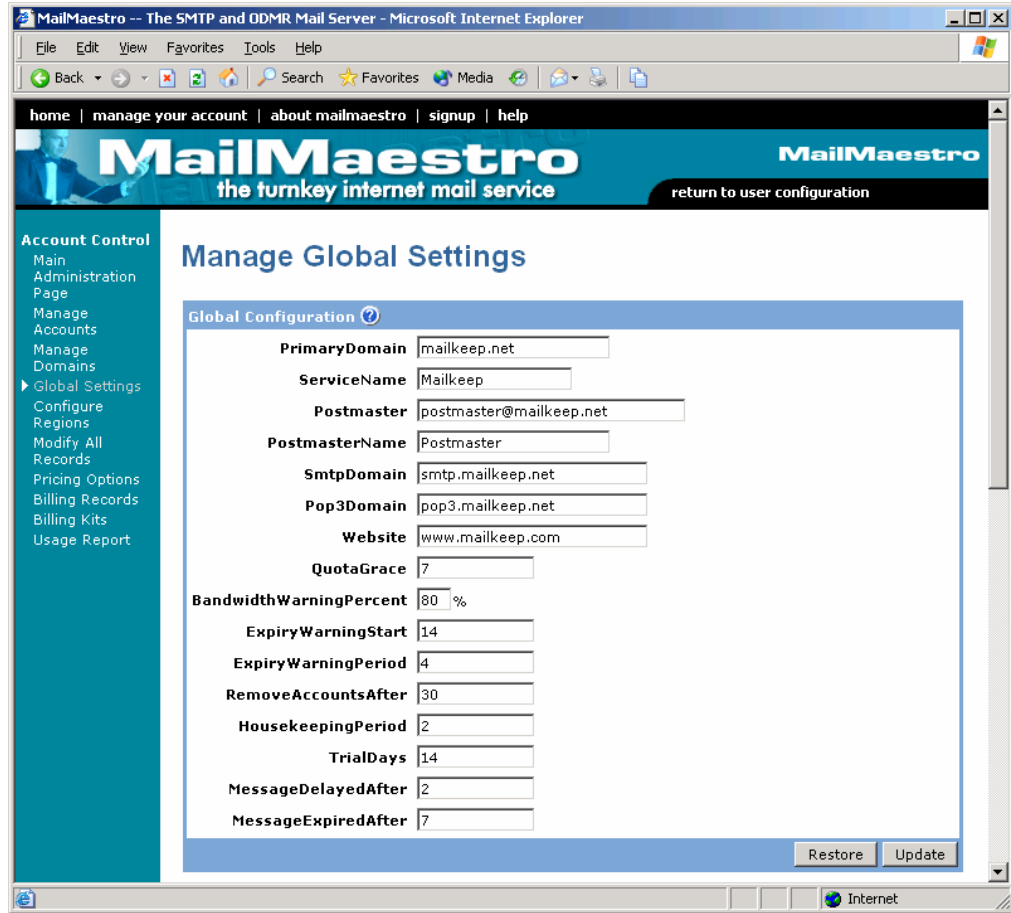


Figure 5 — Administration Site (Global Settings Page)

3. Installing MailMaestro

Installation is a very simple procedure. We have designed the installation procedure to get a real system up and running as quickly as possible. This is because we think you will prefer an iterative configuration process, rather than having to make all the configuration decisions before seeing the system operate. Getting a real system running allows you to see the effect of changes and settings.

You will either have been given a **MailMaestro.msi** file which contains a complete system deployment. The file includes :—

- 1) The MailMaestro engine,
- 2) the middle-tier Web Site API,
- 3) and a complete sample web site which you can customise.

3.1. Requirements

MailMaestro requires one of the following operating systems :—

- Windows 2000 Server
- Windows 2000 Advanced Server
- Windows .NET Server
- Windows .NET Web Server
- Windows .NET Advanced Server

MailMaestro requires IIS (Internet Information Server) 5 or greater.

Note: MailMaestro will work with Windows NT4 Server and Windows NT 4 Enterprise Server, but IIS 4 must be installed and the web site will have to be installed manually.

IIS must be installed and running before you begin installing MailMaestro. During installation MailMaestro will disable any existing web sites and the standard Microsoft Mail Servers (should it be running or installed). This is necessary to allow MailMaestro to operate. You may re-enable the other web sites after installation if you wish.

3.2. Running the Installation

Once you have prepared your machine you can commence the installation.

Note: You must be logged in as an administrator to perform this task.

First run the MailMaestro.msi file. You will go through a series of installation screens. In the Customer Information screen (Figure 6) you should enter your name and the name of your organisation. You should leave the **Anyone who uses this computer (all users)** option checked unless you wish to hide the links to the management pages from other users.

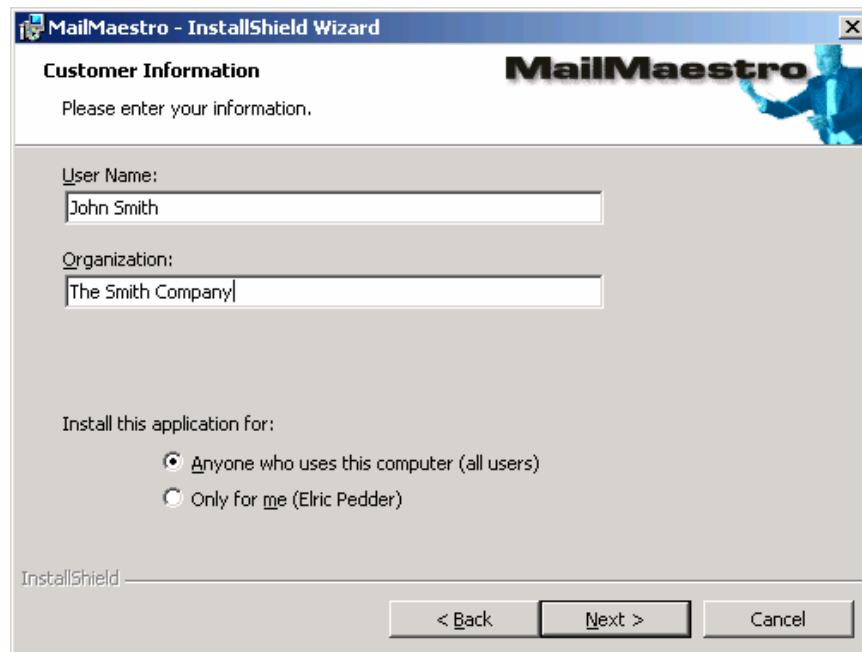


Figure 6 — Customer Information Screen

You should select the **Complete** option unless you want to change the location in which MailMaestro is installed. You should not change the

location unless you have to install MailMaestro on a different drive to the default one (as other documentation may make reference to the default path).

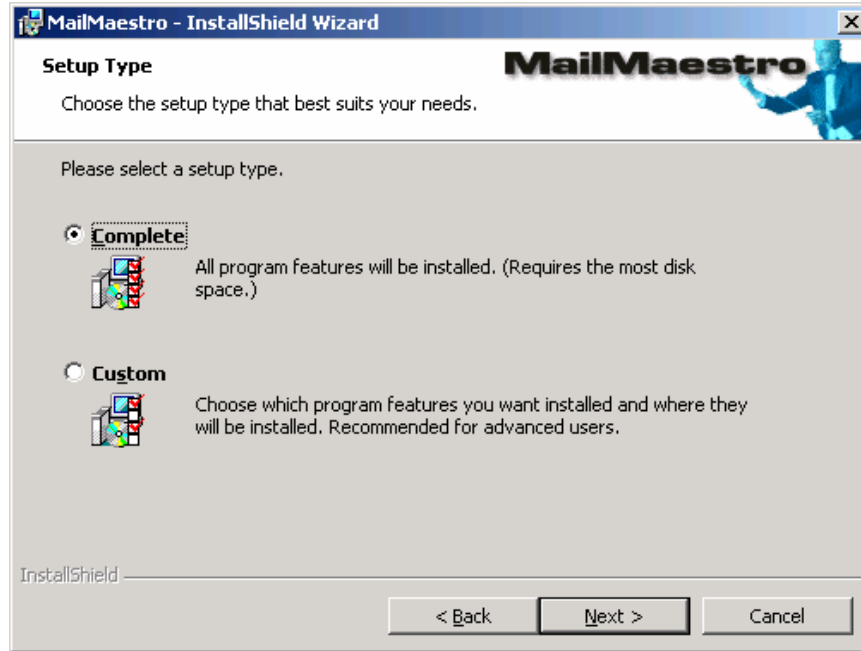


Figure 7 — Installation Type

If you wish to change the installation path, check **Custom** and then **Next**. On the next screen (shown only if **Custom** is checked) you can change the path by clicking on **Change**.

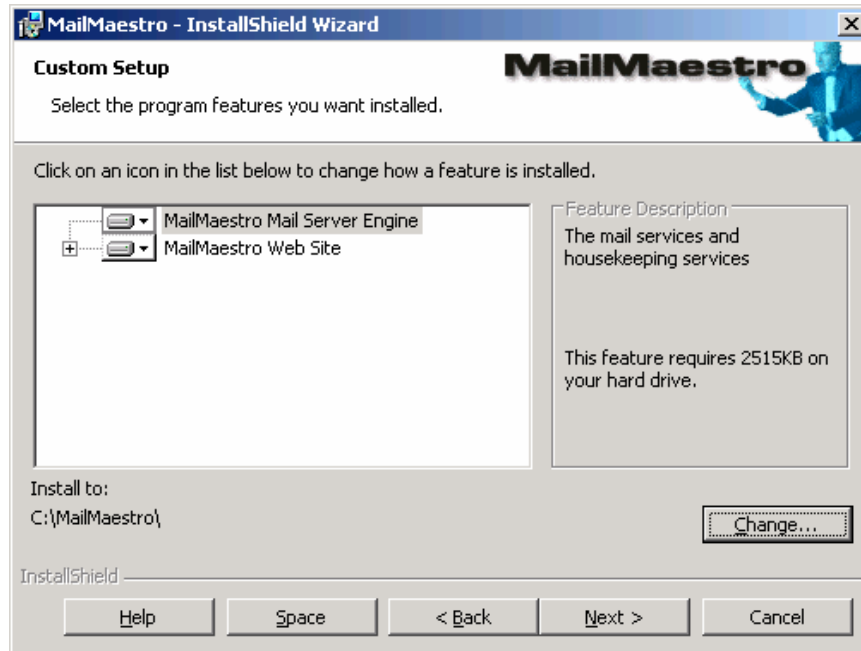


Figure 8 — Changing the Installation Path

Once you have completed this screen you can commence installation by clicking on Install.

4. Configuring MailMaestro

MailMaestro is quite straightforward to configure and you will find that most often defaults are quite appropriate. The following steps represent the basic and necessary configuration :—

- Global Configuration Settings
- Region Configuration
- Pricing Options
- Billing Configuration

Once these configuration steps have been taken, MailMaestro will be operational. You will probably want to modify the template pages to suite your organisation's style.

4.1. The Administration Interface

To access the Administration interface, browse to <http://127.0.0.1/admin> on the MailMaestro machine (or replace the 127.0.0.1 with the name of your machine to remotely administer the system).

You will be asked to log in (and if not, check the security settings in IIS) and you should give the credentials of a user on the machine.

4.2. Global Configuration

Click on “Global Settings” to edit the Global Configuration settings. You will see a screen similar to Figure 9.

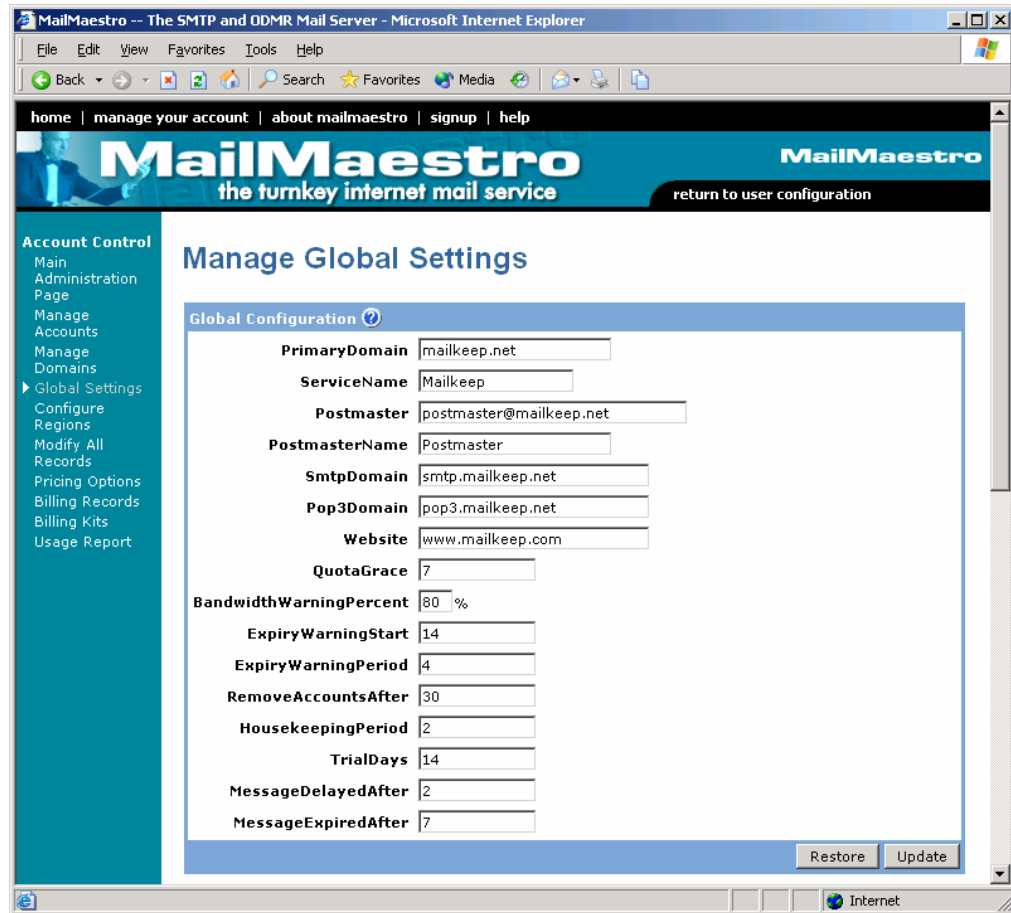


Figure 9 — The Global Settings Page

You should change all the domain names in this page to match your new service — though keep the format similar to the defaults.

Refer to the table on page 29 for a description of the settings.

4.2.1. The Template Account

Scrolling down this page you will see Template Configuration. These settings will be applied to every newly created account. You should set the quota and monthly bandwidth (measured in kilobytes) to a suitable starting amount, and specify the number of domains users should have in each account.

It is important to set the number of domains to the lowest number you will offer in your pricing options as accounts will operate with excess domains, users are simply prevented from creating domains beyond their limit.

Refer to the table on page 34 for additional details.

4.3. Region Configuration

Click on Configure Regions to display the region page. A region is simply a category that user accounts are assigned to — they do not need to represent geographical regions. In practice regions provide a means to determine which pricing options a user can access and what tax rate they must pay.

When a user first signs up, they must select their region.

Each region has an ID, Description and Tax rate. The ID should be unique but you can re-use the description as this is what the user will see. If a region is not marked “Visible” then when a user signs up for a new account, they won't be able to select the region. The administrator, however, can assign a user to any region — regardless of its visibility.

Region descriptions can contain a slash, in which case they are presented in a hierarchy. Some example regions are shown in Figure 10.

The screenshot shows the 'Configure Regions' page in a Microsoft Internet Explorer browser. The page title is 'MailMaestro -- The SMTP and ODMR Mail Server - Microsoft Internet Explorer'. The browser's address bar shows the URL. The page content includes a navigation menu on the left with options like 'Account Control', 'Main Administration Page', 'Manage Accounts', 'Manage Domains', 'Global Settings', 'Configure Regions', 'Modify All Records', 'Pricing Options', 'Billing Records', 'Billing Kits', and 'Usage Report'. The main content area is titled 'Configure Regions' and contains a table of 'Current Regions'.

| ID | Description | Tax | Visible |
|-----|--|-------|---|
| UK | Europe/United Kingdom | 17.5% | Visible <input checked="" type="checkbox"/> |
| EU | Europe/Rest of Europe | 17.5% | Visible <input checked="" type="checkbox"/> |
| ROW | Rest of World | 0% | Visible <input checked="" type="checkbox"/> |
| NL | Europe/Netherlands | 0% | Visible <input type="checkbox"/> |
| US | North America/United States of America | 0% | Visible <input type="checkbox"/> |
| CA | North America/Canada | 0% | Visible <input type="checkbox"/> |
| MX | North America/Mexico | 0% | Visible <input type="checkbox"/> |
| AU | Australia | 0% | Visible <input type="checkbox"/> |

At the bottom of the table, there are three buttons: 'Reset', 'Delete Selected', and 'Add Region'.

Figure 10 — Region Settings

Click on “Add Region” to create a new region record, and be sure to change the ID, Description and Tax rate and then click “Apply Changes” below (otherwise your settings won't be updated).

On the signup page, these regions are displayed in the manner shown in Figure 11. Notice how all the descriptions that began with “North America/” are grouped into a single drop-down box.

Where are you located?

Europe

Rest of World

North America

Australia

United States of America
Canada
Mexico

Figure 11 — Presenting Regions to the User

By hiding certain regions, you can use them to provide special offers to users. You can always change a user's region in their Account settings (reached from the Manage Accounts option on the left).

4.4. Pricing Options

You will probably want to offer users a choice of accounts. Click on “Pricing Options” to configure the available options. Figure 12 shows an example configuration.

To add a new pricing option, simply complete the form titled “New Option” and click “Add”. To edit an option, click on its Name and then edit the form titled “Editing Options ...” before clicking on the “Update” button.

The “Months” value refers to the number of months of access this option provides. The “Bandwidth” value refers to the quantity of data that can be received in the account every month.

The “Quota” refers to the volume of mail that can be stored in the account. In Global Configuration, you specified the Quota Grace period. This is number of consecutive days that the user can exceed this value. After that period the account becomes locked.

Check all the regions in which this pricing option should be offered. The administrator can always manually assign a pricing option to an account manually.

4.4.1. Excess Charges

If you check the “Excess Usage Charge”, then the pricing option will refer only to bandwidth beyond the standard the basic limit. The charge is applied in blocks specified by the bandwidth value. If the settings are \$50 per 4096kb (4mb) and the user goes over their bandwidth by 5000kb, then they will be charged \$100.

In regular pricing options you can optionally select an excess option. If you don't then the user will not be permitted to exceed that limit. (When they do, the account will not accept additional mail.)

The screenshot shows the MailMaestro Administrator's Guide interface. The main content area is titled "Manage Pricing Options". It features a table of "Current Options" and a "New Option" form.

| Name | Price | Months | Quota (kb) | Domains | Bandwidth (kb/m) | Billing Kit | Description |
|--------|---------|--------|------------------------|---------|------------------|-------------|--|
| UK-STD | 35 GBP | 12 | 10240 | 10 | 40960 | worldpay | Standard Option <input type="checkbox"/> |
| UK-HC | 100 GBP | 12 | 102400 | 10 | 409600 | worldpay | High Capacity <input type="checkbox"/> |
| UK-EX | 15 GBP | | (Excess Charge per kb) | | 4 | worldpay | Excess Charge <input type="checkbox"/> |
| A000 | 50 USD | 12 | 10240 | 10 | 40960 | worldpay | Standard Option <input type="checkbox"/> |

The "New Option" form includes the following fields:

- Name: A000
- Price: 50 USD (US Dollar)
- Excess Usage Charge:
- Months: 12
- Quota (kb): 10240
- Domains: 10
- Bandwidth (kb/m): 40960
- Description: Standard Option
- Billing Kit: -- default --
- Regions:
 - Europe/United Kingdom
 - Europe/Rest of Europe
 - Rest of World
 - Europe/Netherlands
 - North America/United States of America
 - North America/Canada
 - North America/Mexico
 - Australia
- Bandwidth Excess Option: (Hard Limit)

Figure 12 — Configuring Pricing Options

4.5. Billing Configuration

The last stage of your configuration is to prepare your system to accept payment online, or to send payment data to you.

Click on “Billing Kits” on the left and you will see the available billing kits. You should click on the billing kit name to edit the details. Additional billing kits will be available from your reseller or from the support web site.

One item of particular importance is the Invoice Template, which is stored in the “templates” directory. This is an invoice that the user can print out. You may wish to customise one to show your own logo and company details.

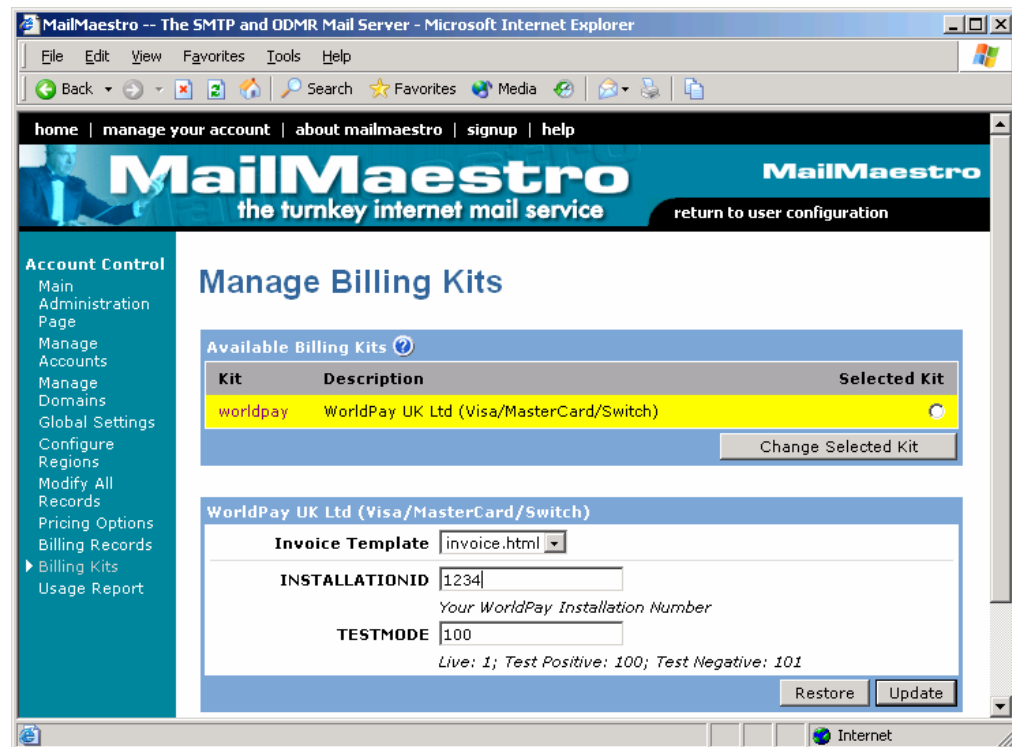


Figure 13 — Billing Kit Configuration

When a user purchases a price option an invoice is created and stored in their account. A global billing record is also maintained and contains links to each

of these invoices, grouped according to month. You can see these billing records from the administration section.

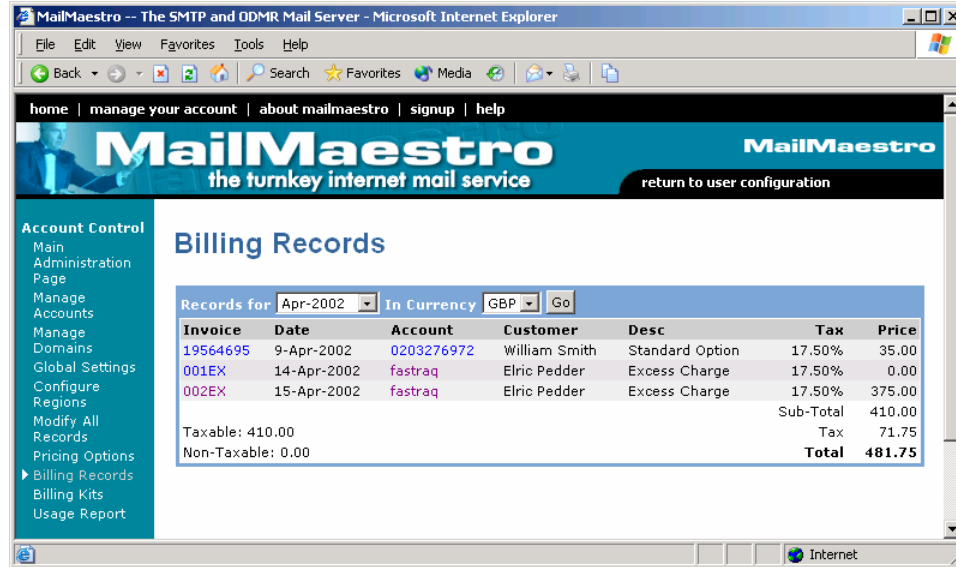


Figure 14 — Viewing Billing Records for a month

5. MailMaestro Implementation

MailMaestro consists of two parts :-

- The Mail Server (NT Services and Client)
- The Administrative/Management Interfaces and Web Sites

5.1. The Mail Server

The server consists of three NT Services (**MailMaestroSMTP**, **MailMaestroPOP3** and **MailMaestroHousekeeping**) and the mail delivery client.

5.1.1. MailMaestroSMTP Service

The **MailMaestroSMTP** server receives incoming mail for the customer, and also allows the customer to connect via SMTP to make ETRN requests, and via the ODMR port.

The service listens on the following TCP/IP ports :—

- **25** (SMTP) — For incoming mail, ODMR and ETRN
- **366** (ODMR) — This is the ODMR specific port
- **2525** — This alternative port allows users to make ETRN or ODMR requests when their service provider blocks outbound connections on known ports.

You can start and stop this service from the command line using the Windows NT **net** utility :—

```
net stop MailMaestroSMTP
net start MailMaestroSMTP
```

Transactions that do not complete, or which involve an error, are logged in the file **etranscript.txt** (which is in the same directory as the service).

5.1.2. MailMaestroPOP3 Service

The **MailMaestroPOP3** server provides POP3 mailbox access to clients. MailMaestro provides an advanced POP3 server with a number of extensions :—

- Demon Internet style *ENV commands (supported by Mailtraq)
- Virtual Mailboxes requested by **user@account**, **user@domain** and **@domain**.
- Secure Authentication using APOP
- Secure sub-mailboxes with separate MD5 derived passwords

The virtual mailboxes allow multiple users to share a single MailMaestro account. When logging on in POP3 the client sends a “username” and “password”. That username can be the MailMaestro account number, or a virtual mailbox (e.g. **user@domain** which creates a virtual mailbox containing only mail addressed to that this specification).

The **MailMaestroPOP3** server listens on TCP/IP port 110 (POP3). Once the client has authenticated, a transcript of their session is kept and when the user disconnects that transcript is appended to the account holder's daily log file.

5.1.3. MailMaestroHousekeeping Service

This service performs periodic housekeeping services. These services include :—

- **Scanning for expired accounts, or accounts nearing expiry.**
When such an account is found an email is inserted into the account warning the account holder.
- **Scanning for messages that have been waiting for delivery for too long.**
A message is considered “waiting” if it arrives in the account after the last account access time. After a message has been waiting for the period defined by **MessageDelayedAfter** (Global Settings) a Delivery Report (Delayed) message is returned to the return-path (if it is not empty). When the message has been waiting for the period defined by **MessageExpiredAfter** (Global Settings) a Delivery Report (Failure) is returned and the message

is deleted. This is a necessary step to ensure that the sender of the message is aware that their message was not delivered in a timely fashion.

- **Removing Expired Accounts.**
An account, including all mail, logs and settings, will be erased **RemoveAccountsAfter** days after expiry.
- **Scanning for orphan messages.**
Should some problem occur whereby either the **.env** or **.dat** part of a message is lost, an orphan report will be prepared for the account owner. This typically occurs if the service is shut down during mail receipt, although as SMTP is a transacted protocol it almost never represents a truly "lost" message.

The period between housekeeping operations is defined in the Global Settings.

5.1.4. Mail Delivery Client

The mail server also includes **MailMaestroClient**, which is an SMTP delivery process that is capable of sending all or some of the mail stored in an account to a destination IP address using SMTP. This process is only fired on demand, as a result of an ETRN trigger or the arrival of new mail (when direct SMTP is enabled).

5.2. Administrative Interface

This part of MailMaestro will be what the customers see when they apply for an account and manage it. As a result, it is designed to be highly customizable. We provide a template interface as part of the MailMaestro package, and almost everything in it can be altered.

The interface is provided primarily through Internet Information Server (part of the Windows Server deployment) using Active Server Pages (ASP). The heart of MailMaestro's administrative system is **mailmaestro.asp**, which should be included in any page that accesses the system. MailMaestro requires its clients to use JavaScript, but very little must be written to implement an effective interface.

mailmaestro.asp has been designed to simplify management by concentrating all of the system and database access into a defined API. For example,

authentication is performed using `MM_Authenticate()`. The following statement essentially authenticates a session :—

```
Session("account") = MM_Authenticate(  
    Request("account"), Request("password")  
);
```

Account creation is trivial, as this one line example shows :—

```
var account = MM_CreateAccount();
```

6. General Configuration

MailMaestro is configured using basic text files, making it very easy to monitor and modify using regular tools. It also makes it simple to manage through custom software.

6.1. Global Configuration

Global configuration is stored in the file **global.inf**, in the accounts directory.

PrimaryDomain

This is domain name that your service uses. One would expect, for example, that your postmaster address be at this domain name. An example is **mailmaestro.net**.

ServiceName

This is the commercial name of your service (for example, **MailMaestro**). This name will appear in various automatically generated messages.

Postmaster

This is the e-mail address of your service's postmaster. Typically, this is *postmaster@PrimaryDomain*.

PostmasterName

This is the name of the postmaster, typically just *Postmaster*.

SmtpDomain

This is the domain name of your SMTP server. This domain name will be used in the Received: header of messages and in the SMTP banners. Example: **smtp.mailmaestro.net**.

Pop3Domain

This is the domain name of your POP3 server. It will be used in the POP3 connection banner.

| | |
|--------------------------------|---|
| QuotaGrace | This is the number of days that your users are allowed to exceed their quota. If they have not returned within their quota in this period new mail will be refused. |
| BandwidthWarningPercent | This is the percentage of bandwidth used in a month before a warning is sent to the account owner. |
| Website | This is the domain name of your service's web site. It will be mentioned in expiry notices. Example: www.mailmaestro.com . |
| ExpiryWarningStart | This is the number of days prior to expiry before expiry warnings are sent to the account owner. |
| ExpiryWarningPeriod | This is the number of days between expiry warnings. The last warning will be sent after the account has expired. |
| RemoveAccountsAfter | This is the number of days after expiry when the account (including configuration and messages) will be deleted. |
| HousekeepingPeriod | This is the number of hours between housekeeping operations. |
| TrialDays | This is the number of days before expiry assigned to all new accounts. |

7. Account Configuration

This section explains the essential files used in the configuration of the MailMaestro user accounts.

Two directories are involved: **accounts** and **domains**. The **accounts** directory contains sub-directories for each of the active user accounts. The **domains** directory contains a file for every registered domain. That file contains a single line: the account that owns that domain.

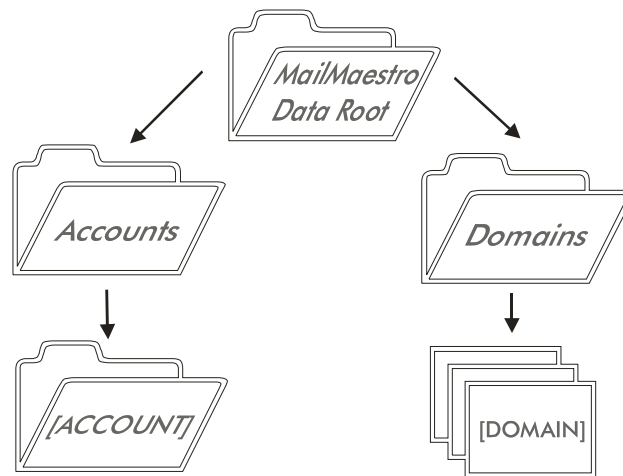


Figure 15 — Directory Structure

In reverse, the account for a user includes a file, **domains.inf**, which lists all the domains owned by that account. The **domains.inf** files and the domains directory must be kept synchronized.

The account directory contains configuration files, log files, invoices, billing records, bandwidth usage records and the messages themselves.

7.1. Accounts Directory Files

The files found in the accounts directory are described below.

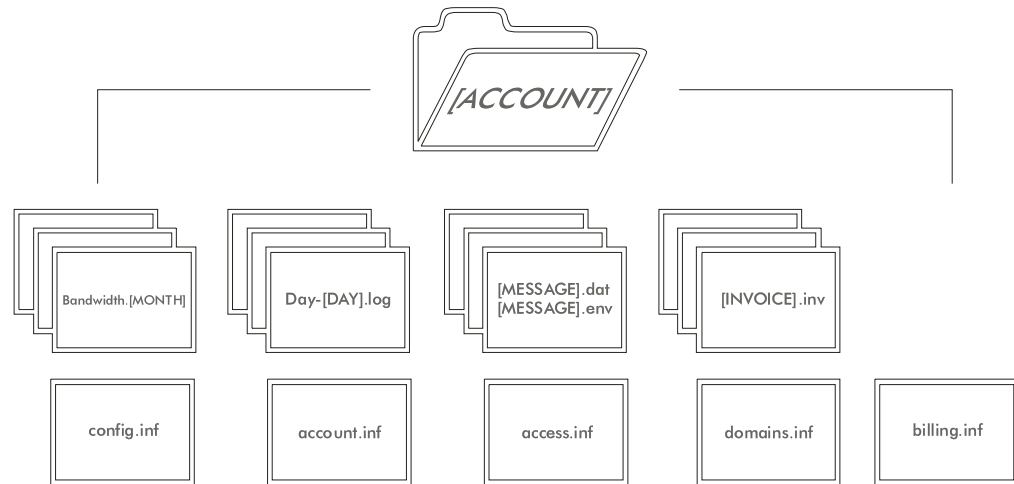


Figure 16 — Account Contents

7.1.1. Message Files (.env and .dat)

Each message stored in the account consists of two files: the message data (.dat) and the message envelope (.env). The same filename is used for both, and this is a globally unique number formed by a time-stamp and a serial number.

The envelope file is a text file that consists of two or more lines. The first line is the sender (the **MAIL FROM:** <> parameter given during the **SMTP** transaction). This is also considered the Return-Path. This line may be blank if no return-path is given.

The second line is the recipient of the message (parameter given by the **RCPT TO:** <> command in the **SMTP** transaction). If the message has multiple recipients, a duplicate .env and .dat file set will be created for each recipient.

An example .env file is given below :—

```
postmaster@mailmaestro.com
recipient@yourcompany.com
```

The .dat file is an exact copy of the received message data, with a single Received: header added (as required by RFC821).

7.1.2. Log Files (Day-*n*.log)

Every time mail is collected from the account (or sent via SMTP) the protocol transaction (POP3 or SMTP) is recorded in a log file in the account. This gives the user the greatest ability to identify problems.

An example log is shown below. Each transaction is separated with a timestamp.

```
----- Sat, 02 Feb 2002 05:59:58 -0000
Connecting to... 204.92.85.2

220 novitraq.com Ready for action (Mailtraq 2.0.0.1242/SMTP)
; could not identify keycheck string: fastraq

----- Sat, 02 Feb 2002 06:23:55 -0000
Connecting by SMTP/TURN to...

220 novitraq.com Ready for action (Mailtraq 2.0.0.1242/SMTP)
HELO smtp.mailmaestro.net
250 novitraq.com
MAIL FROM:<mail@sendfree.com>
250 receiving from mail@sendfree.com
RCPT TO:<john-NOSPAM@novitraq.com>
550 cannot relay for john-NOSPAM@novitraq.com
RSET
250 clearing sender and recipient list, go ahead
QUIT
221 have a nice day (SMTP Closing)
```

This transaction is appended to a file called **Day-*n*.log**, where ***n*** is the current day number. Each file only contains log information for a single day. As a result, up to 31 days worth of log data is kept at any time.

7.1.3. Bandwidth Files (bandwidth.*n*)

These files record the amount of data written to the account in any given month. The filename is **bandwidth.*n***, where ***n*** is the current month number. Each file contains data for a single month, and as a result up to one year of bandwidth data is recorded. Each file consists of a single line with a value representing the number of bytes transferred.

7.1.4. Access Record ([access.inf](#))

This file contains information regarding the last time the account was accessed by its owner.

| | |
|-------------------------|---|
| AccessType | This will contain POP3 or SMTP. POP3 indicates that the owner connected via the POP3 protocol to inspect or retrieve messages. SMTP indicates that the owner either connected via SMTP (and issued an ETRN or ATRN command) or the SMTP connection to the owner was initiated by the receipt of mail. |
| AccessClientHost | This is the reverse DNS lookup of the client's IP |
| AccessClientIP | This is the IP address of the client. This IP address is used in receipt-triggered SMTP connections to the client. |

7.1.5. Configuration ([config.inf](#))

This file contains account configuration information, including account privileges and access.

| | |
|-----------------|--|
| account | This is the account ID |
| password | This is the owner's access password in clear text |
| quota | This is the quota (in kilobytes) assigned to this account. If the contents of the account exceed this amount, the owner will be warned and the grace period will commence. The presence of an empty quota.inf file indicates this grace period has commenced (the file's timestamp indicates when the grace period began). After the grace period, the account will be locked (new mail will be rejected with a 450 SMTP temporary response) and the quota.inf file will contain an RFC822 time stamp indicating when the account was locked. If at any time during the grace period, the account size drops below the quota value the quota.inf file will be deleted and the account will be released from grace period. A value of zero (0) represents an account |

without a quota.

- bandwidthlimit** This is the limit (in kilobytes) of incoming mail that will be accepted every month. When the bandwidth consumption for any month reaches 80% of this limit the owner will be warned. When the bandwidth exceeds this limit new mail will be rejected with a 450 SMTP response (temporary). The file **bandwidth.inf** is added (zero length) when the warning is issued, and a timestamp is written to it when the limit exceeded message is sent. **bandwidth.n** files are added (where x is the number of the current month) containing the number of bytes used to store incoming mail.
- pdate** This is a timestamp (dd/mm/yyyy hh:nn:ss) indicating when a payment was last made on the account. When a payment is made on the account, the **edate** is extended by the appropriate period.
- edate** This is a timestamp (dd/mm/yyyy hh:nn:ss) indicating when the account will expire. After this period, all incoming mail will be refused with a 550 SMTP response.
- cdate** This is the timestamp (dd/mm/yyyy hh:nn:ss) indicating when the account was created.
- access** This is a string that defines what access methods are permitted for the account. It may include the following characters :—
- A** Access by POP3 but authentication uses the APOP command which uses a cryptographically secure hash (MD5) in combination with a unique id (including timestamp) and the account password. The password is never transmitted and the derivation cannot be used to recreate the password or to repeat the login sequence at another time.
 - E** Access by reverse SMTP (ETRN) whereby MailMaestro will initiate a connection to the client on the SMTP channel and deliver waiting mail.

- P Access by POP3 using a clear-text password.
- O Access by ODMR, which is similar to ETRN but does not involve the initiation of a separate connection to the client — the SMTP roles are simply reversed.
- S Access by recipient-triggered SMTP. When mail arrives, MailMaestro will initiate a connection by SMTP to the last IP address that connected and authenticated. Must be used in conjunction with one of the other methods.

excessoption If this is "0" (or blank) then the bandwidth limit is hard. Otherwise, this will refer to an excess pricing option and the account may exceed its bandwidth limits. At the end of the month **MailMaestroHousekeeping** will bill the user based on the *excessoption* set at that time.

7.1.6. Account Information (**account.inf**)

This file contains general information about the account holder.

| | |
|-----------------------|--|
| Email | This is a contact e-mail address of the owner that will not go through MailMaestro. |
| AdminEmail | This is the user part of the MailMaestro managed contact e-mail address |
| AdminFullEmail | This is a fully qualified contact e-mail address which matches a domain inside the MailMaestro account |
| AccountName | This is the name by which the account is referred to, and would typically be the company name or (if the customer is an individual) the customer's full name |
| Title | General Salutation Prefix (e.g. Ms, Dr) |
| Firstname | User's First Name |
| Lastname | User's Last Name |
| Address | The user's postal contact address. Lines should be |

separated by

| | |
|---------------------|---|
| Postcode | The user's postal code / ZIP code |
| Country | Generally this will be the ISO country code compatible with the billing kit |
| Telephone | User's telephone number |
| Fax | User's fax number |
| CreationDate | The date on which the customer record was created |
| Region | The region code taken from the region table |

8. MailMaestro Web Site API

The middle-tier API is contained in the file **mailmaestro.asp** which should be included in your user section using the following directive :—

```
<!--#include file="admin/mailmaestro.asp"-->
```

It should be included in the admin section using :—

```
<!--#include file="mailmaestro.asp"-->
```

8.1. Types

This section describes the types and objects that are commonly used in this API.

8.1.1. Dates

There are two formats for dates. The internal JavaScript Date object, and the MailMaestro date string. The Date object has various member functions that are useful for performing date based comparisons and calculations. Unfortunately there is no standard string representation of the date, which is why we also use MailMaestro dates.

These are simply strings that are either displayed as **dd/mm/yyyy hh:mm:ss** or just **dd/mm/yyyy**. If the time part of the date is excluded, it is assumed to be **00:00:00**. An example is **31/12/2002 15:59:45**.

Conversions between these formats are achieved with **MM_EncodeDate()** (which converts a Date object to a MailMaestro date string) and **MM_DecodeDate()** (which performs the reverse).

8.1.2. Arrays

When a list of items needs to be returned, MailMaestro uses JavaScript arrays. These are zero indexed and can contain any type of object.

8.1.3. Inf Files

It is often necessary to load configuration files (such as the config.inf or account.inf files). These are stored in an Inf format, which is essentially a table of named properties. Use the functions **MM_GetProp()** and **MM_SetProp()** to access the properties in the table. For example,

```
var inf = MM_LoadConfig("01234567");
var firstname = MM_GetProp(inf, MM_CONFIG_FIRSTNAME)
```

8.2. Support Functions

These functions provide general facilities.

8.2.1. MM_StringsEqual

```
bool = MM_StringsEqual(s1, s2)
```

Returns **true** if the string representations of **s1** and **s2** are the same (case sensitive).

8.2.2. MM_SetDecimalPlaces

```
float = MM_SetDecimalPlaces(value, decimals)
```

Returns the value limited to the given number of decimal places. For example, **MM_SetDecimalPlaces(12.67, 1)** would return 12.7.

8.2.3. MM_FormatFloat

```
string = MM_FormatFloat(value, decimals)
```

Returns a string containing the given value but with a fixed number of decimal places. For example, **MM_FormatFloat(12.67, 3)** would return 12.670.

8.2.4. MM_FormatBytes

```
string = MM_FormatBytes(value)
```

Returns a string containing the data size representing the value (in bytes). An appropriate byte unit is selected for the value. For example, **MM_FormatBytes(102400)** would return "100kb".

8.2.5. MM_RFC822TimeStamp

```
string = MM_RFC822TimeStamp(date)
```

Returns a date in the RFC822 format (as it appears in message Date: fields)

date The date for which a time stamp is to be created

8.2.6. MM_AccountPath

```
account = MM_AccountPath(account)
```

Returns the filing system path to the given account (ending with a backslash)

account The account number for which a path is to be returned

8.2.7. MM_DomainToAccount

```
account = MM_DomainToAccount(domain)
```

Returns the account number that owns the given domain, or null if the domain is not assigned. This function will traverse the domain hierarchy to find matching accounts. For example, if account **01234567** owns **domain.com**, then calling **MM_DomainToAccount("my.domain.com")** will return **01234567**.

domain This is the domain string to search for. It can also be an account number.

8.2.8. MM_AccountExists

```
bool = MM_AccountExists(account)
```

Returns **true** if the account number (or domain) is valid

account The account number (or domain) to verify

8.2.9. MM_GetProp

```
string = MM_GetProp(Inf, prop)
```

Returns the property named prop in an Inf table.

8.2.10. MM_SetProp

```
MM_SetProp(inf, prop, value)
```

Stores the value named `prop` in an Inf table. If the property already exists in the table, it is replaced by the new value.

8.2.11. MM_DeleteProp

```
MM_DeleteProp(inf, prop)
```

Removes a property named `prop` from the Inf table.

8.2.12. MM_ListGetKeys

```
array = MM_ListGetKeys(inf)
```

Returns an array containing all the property names in an Inf table.

8.2.13. MM_ListGetItems

```
array = MM_ListGetItems(inf)
```

Returns an array containing all the property values in an Inf table.

8.2.14. MM_ListSortItems

```
array = MM_ListSortItems(inf, itemtype)
```

Returns an array of index values pointing to properties in the Inf table sorted according to `itemtype`. If `itemtype` is "n" the properties are treated as numbers. If `itemtype` is "d" the properties are treated as MailMaestro dates. If `itemtype` is not given, the properties are treated as strings.

Use the resulting array as an index to the array returned by `MM_ListGetItems()` and `MM_ListGetKeys()`.

8.2.15. MM_ZeroLead

```
string = MM_ZeroLead(number, width)
```

Returns a string representing the given number, left padded with zeros to reach the required width. For example,

```
var str = MM_ZeroLead(123, 6);
return str == "000123";
```

8.2.16. MM_VerifyDate

```
bool = MM_VerifyDate(str)
```

Returns true if the MailMaestro date string str is both valid and properly formatted.

8.2.17. MM_EncodeDate

```
string = MM_EncodeDate(date)
```

Returns the MailMaestro string format of the given date

8.2.18. MM_DecodeDate

```
date = MM_DecodeDate(string)
```

Returns the JavaScript Date object represented by the MailMaestro string.

8.2.19. MM_StripTimeFromDate

```
string = MM_StripTimeFromDate(datestring)
```

This function takes a MailMaestro date string and strips the time part (if it exist) returning only the date part.

8.3. Configuration Files

These functions provide access to the global configuration settings and to individual account configuration files.

8.3.1. MM_LoadGlobalConfig

```
inf = MM_LoadGlobalConfig()
```

Returns the Inf table containing the global configuration settings. The available properties are **MM_GCONFIG_XXX**.

8.3.2. MM_StoreGlobalConfig

```
MM_StoreGlobalConfig(inf)
```

Stores a (modified) Inf table containing the global configuration settings. You should only store Inf files created with `MM_LoadGlobalConfig()`.

8.3.3. MM_LoadConfig

```
inf = MM_LoadConfig(account)
```

Returns an Inf table containing the configuration settings for the specified account. This function will return an empty table if the file does not exist. The available properties are `MM_CONFIG_xxx`.

8.3.4. MM_StoreConfig

```
MM_StoreConfig(inf, account)
```

Stores a (modified) Inf table containing the account configuration settings. You should only store Inf files created with `MM_LoadConfig()`.

8.3.5. MM_LoadAccount

```
inf = MM_LoadAccount(account)
```

Returns an Inf table containing the account information for the specified account. This function will return an empty table if the file does not exist. The available properties are `MM_ACCOUNT_xxx`.

8.3.6. MM_StoreAccount

```
MM_StoreAccount(inf, account)
```

Stores a (modified) Inf table containing the account information. You should only store Inf files created with `MM_LoadAccount()`.

8.3.7. MM_LoadAccess

```
inf = MM_LoadAccess(account)
```

Returns an Inf table containing the account access information for the specified account. This function will return an empty table if the file does not exist. The available properties are `MM_ACCESS_XXX`.

8.3.8. MM_StoreAccess

```
MM_StoreAccess(inf, account)
```

Stores a (modified) Inf table containing the account access information for the specified account. You should only store Inf files created with `MM_LoadAccess()`.

8.3.9. MM_ExtendAccountExpiry

```
MM_ExtendAccountExpiry(account, quantity, unit)
```

Extends the expiry date of a given account by the specified time

- account** The account number that should be updated
- quantity** The number of units to extend the expiry by
- unit** The unit of time ("D" for Day, "M" for Month or "Y" for Year)

8.4. Log Functions

These functions provide access to a MailMaestro account's log files. Log files are referred to by date which represents the day the log file was created for.

8.4.1. MM_GetLogList

```
array = MM_GetLogList(account)
```

Returns an array of log dates available for the given account.

8.4.2. MM_CreateLogMatrix

```
array = MM_CreateLogMatrix(logarray)
```

Takes an array returned by `MM_GetLogList()` and spreads it out into a new array of dates indexed by day. The first item in the array will always be a Sunday, and each successive element will be one day later. If a log is not available for that day, the entry is a null.

8.4.3. MM_GetLog

```
text = MM_GetLog(account, date)
```

Given a date (JavaScript Date object taken from `MM_GetLogList()` or `MM_CreateLogMatrix()`) and account number, returns a text file containing the log data for that day. Returns null if no log for that day is available.

8.5. Account Management Functions

These functions provide general management for MailMaestro accounts.

8.5.1. MM_ConfigAccessEnabled

```
bool = MM_ConfigAccessEnabled(inf, access)
```

Returns true if the Inf table (generated by `MM_LoadConfig()`) has the specified access method enabled. Refer to the table on page 34 for a list of the access methods.

8.5.2. MM_ConfigChangeAccess

```
MM_ConfigChangeAccess(inf, access, enable)
```

Either enables or disables a given access method in the given Inf table (generated by `MM_LoadConfig()`). Enable should be true or false.

8.5.3. MM_Authenticate

```
bool = MM_Authenticate(account, password)
```

This function performs web-site authentication. Given an account name (or domain) and password, the function will check that the password matches that in the account (with case sensitivity).

8.5.4. MM_GetAccountList

```
array = MM_GetAccountList()
```

Returns an array of all the accounts (by number) configured in the system. These may include expired accounts.

8.5.5. MM_CreateAccount

```
string = MM_CreateAccount()
```

Creates a new account and returns the new account number. The account configuration is copied from the “_template” account.

8.5.6. MM_BulkGetProp

```
inf = MM_BulkGetProp(list, store, prop, start, max, inf)
```

This function gets a single specified property from every account in the given list and returns it in an Inf table.

| | |
|--------------|--|
| list | An array of accounts from which the given property is to be extracted |
| store | The Inf table from which the property is to be extracted. Can be MM_FILE_CONFIG , MM_FILE_ACCOUNT , MM_FILE_ACCESS or the special values MM_BULK_QUOTA (returns the amount of space used for each account) or MM_BULK_ADATE (returns the last time the account was accessed). |
| prop | The property to extract (not used for MM_BULK_QUOTA or MM_BULK_ADATE) |
| start | (Optional) The first item in the list to be returned |
| max | (Optional) The maximum number of items to return |
| inf | (Optional) The Inf table in which properties are to be added. If not given, a new Inf table will be created. |

8.5.7. MM_BulkChangeProp

```
int = MM_BulkChangeProp(list, store, prop, oldvalue, newvalue)
```

Changes all the properties in the **list** named **prop** in **store** that match **oldvalue** (if **oldvalue** is not null) to **newvalue**.

8.5.8. MM_GetBandwidthUsed

```
int = MM_GetBandwidthUsed(account, month)
```

Returns the number of kilobytes of bandwidth consumed by the given account in the given month (a JavaScript Date object).

8.6. Message Access

These functions provide access to the messages stored in a MailMaestro account.

8.6.1. MM_GetMessageSizeTotal

```
int = MM_GetMessageSizeTotal(account)
```

Returns the total number of kilobytes occupied by messages in the given account.

8.6.2. MM_GetMessageData

```
inf = MM_GetMessageData(account, msg, inf)
```

Returns an Inf table populated with properties regarding the given message. If inf is not given, a new table will be created. These properties are given below :—

MM_MSG_SENDER

The sender as specified in the message envelope

MM_MSG_RECIPIENTS

A list of recipients (separated by newlines) as specified in the message envelope

MM_MSG_HEADER

The message header (in a single string)

MM_MSG_BODY

The message body (up to 100 lines)

MM_MSG_SIZE

The size (in bytes) of the message

MM_MSG_DATE

The date the message was received (in JavaScript Date format)

8.6.3. MM_DeleteMessage

```
bool = MM_DeleteMessage(account, msg)
```

Removes the given message from the given account, returning true if successful.

8.6.4. MM_GetMessageList

```
array = MM_GetMessageList(account)
```

Returns an array of the message ids that exist in the given account.

8.7. Domain Management

These functions manage the domains in an account. The domains in an account are stored in a file called “domains.inf”, but each account is also cross-referenced in the **domains** directory. Always use these functions as they keep the two items synchronised.

8.7.1. MM_GetAccountDomains

```
array = MM_GetAccountDomains(account)
```

Returns an array of domains configured for the given account number.

8.7.2. MM_AddAccountDomain

```
MM_AddAccountDomain(account, domain)
```

Adds the given domain to the account. No verification is done so you must first ensure that the domain does not already exist or clash with another domain (using **MM_ValidateNewAccountDomain()**).

8.7.3. MM_DeleteAccountDomain

```
MM_DeleteAccountDomain(account, domain)
```

Removes the given domain from the given account.

8.7.4. MM_ParentDomainOf

```
string = MM_ParentDomainOf(domain)
```

Returns the parent (superior) domain of the given domain. For example,

```
var domain = MM_ParentOf("my.domain.com");
return domain == "domain.com";
```

8.7.5. MM_ValidateNewAccountDomain

```
string = MM_ValidateNewAccountDomain(domain)
```

Returns an error message if the domain is unavailable, contains invalid characters or does not appear to be a domain name. Returns null if the domain is OK.

8.7.6. MM_GetAllDomains

```
array = MM_GetAllDomains()
```

Returns an array containing all the domains registered in the system.

8.8. Sending Messages

Often you will want to send messages to MailMaestro account holders. These are e-mail messages sent either directly into their account (for their collection) or via the Internet to the address they have given.

8.8.1. MM_PostMessage

```
MM_PostMessage(account, subject, msg, from, to)
```

Inserts a message into the account specified. The **msg** text may contain macros in the form **#store.property#**. For example, **#ACCOUNT.FIRSTNAME#** or **#CONFIG.EDATE#**.

account The account in which to insert the message

subject The subject line of the new message

msg The message body

- from** The From: field string to use (if not given, defaults to the postmaster name and address)
- to** The message envelope recipient. If not given, the account firstname, lastname and administration e-mail address will be used.

8.8.2. MM_PostMessageExternal

```
MM_PostMessageExternal(account, subject, msg, from, to)
```

Exactly the same as `MM_PostMessage()`, except the message is sent using the Internet (by inserting the message into the outbound mail spool).

8.9. Price Lists

These functions are used to manage the price lists and billing/invoicing systems.

8.9.1. MM_LoadPricelist

```
array = MM_LoadPricelist(region, noexcess)
```

Returns a pricelist array containing the available pricing options applicable to a given **region** (or all regions if **region** is null). If **noexcess** is true, then the pricelist returned will not include the excess charge options.

The array returned is actually a compound array of pricing options, so you should use the other functions to manipulate this data.

8.9.2. MM_GetPriceOption

```
array = MM_GetPriceOption(option)
```

Returns the given pricing option as an array of values. Use the `MM_PRICELIST_XXX` constants as array indices to access the properties. For example :—

```
var option = MM_GetPriceOption("basic");
var price = option[MM_PRICELIST_PRICE];
```

8.9.3. MM_StorePricelist

```
MM_StorePricelist(pricelist)
```

Stores a (modified) price list originally retrieved with `MM_LoadPricelist()`.

8.9.4. MM_AddPricelistOption

```
MM_AddPricelistOption(pricelist, name, price, months, quota,
bandwidth, domainlimit, currency, description, billingkit,
regions, excessoption)
```

Adds a new pricing option to the price list (originally retrieved with `MM_LoadPricelist()`).

- pricelist** The price list returned by `MM_LoadPriceList()`
- name** The key for this pricing option which should be short and unique.
- price** This is the price for the option (specified as a floating point number). For excess options, this is multiplied by the number of bandwidth units to get the charge.
- months** The number of months this option will give to the account. This should be zero for an excess option.
- quota** The number of kilobytes of quota included in the pricing option.
- bandwidth**
The number of kilobytes of incoming data permitted per month in this option. For excess options, this is the unit bandwidth when measuring the price.
- domainlimit**
This is the number of domains included in the account.
- currency** This is the ISO three-letter currency code for this option.
- description**
A string describing the pricing option (does not need to be unique)

billingkit The key of the billing kit to use when purchasing this option.

regions This is a list of regions separated by pipe characters (the vertical bar | character).

excessoption

If this option is not an excess option, it can refer to another excess option key. If it is blank, then the bandwidth limit cannot be exceeded.

8.9.5. MM_DeletePricelistOption

```
MM_DeletePricelistOption(pricelist, option)
```

Removes the identified price list option from the price list.

8.9.6. MM_GetNewPurchaseExpiry

```
date = MM_GetNewPurchaseExpiry(account, priceoption)
```

This function returns the new expiry for the given account if the given **priceoption** were to be applied to it. This allows you to tell the user what their new expiry would be if they purchased a pricing option.

8.9.7. MM_PurchaseOption

```
MM_PurchaseOption(account, priceoption)
```

This function applies the given **priceoption** to the given account. It updates the configuration to reflect the settings specified in the **priceoption**, including extended the expiry. If the specified **priceoption** is an excess charge, then the current excess charges are cleared.

8.10. Billing System

The following functions provide access to the billing system and the billing records.

8.10.1. MM_AddBillingRecord

```
MM_AddBillingRecord(account, invoiceid, transid, priceoption, taxrate, authmsg, invoicetemplate, cust_name, cust_address, cust_email, cust_tel, cust_fax)
```

This function creates a set of billing records and an invoice to represent the purchase of the given **priceoption**. The invoice is filed in the customer's account.

- account** The customer account number
- invoiceid** An arbitrary unique id used to identify this invoice
- transid** An arbitrary transaction id which should refer to the financial transaction involved — it may be the same as the **invoiceid**
- priceoption**
The priceoption being purchased
- taxrate** A floating point value representing the tax paid which is typically taken from the customer's region.
- authmsg** An arbitrary text message typically returned from the financial institution verifying the transaction has been authorised
- invoicetemplate**
The name of an invoice template (from the templates directory) to use when building the invoice for the customer.
- cust_name, cust_address, cust_email, cust_tel, cust_fax**
These values are inserted into the invoice and are used to update the customer account information.

8.10.2. MM_GetBillingRecordMonths

```
array = MM_GetBillingRecordMonths()
```

This function returns an array of filenames representing the months for which billing records are available.

8.10.3. MM_GetBillingGlobal

```
array = MM_GetBillingGlobal(file)
```

This function returns an array of billing records in the specified file (one of the items in the array returned by **MM_GetBillingRecordMonths()**). The array returned contains arrays, each representing a billing record. The sub-arrays are indexed with the **MM_BILLING_xxx** constants. For example :—

```
var months = MM_GetBillingRecordMonths();
var file = MM_GetBillingGlobal(months[0]);
var total_for_month = 0;
for (var i = 0; i < file.length; i++) {
    var record = file[i];
    total_for_month += record[MM_BILLING_PRICE];
}
```

8.10.4. MM_GetBillingForAccount

```
array = MM_GetBillingForAccount(account)
```

This function is similar to `MM_GetBillingRecords()` except it returns all the record for a given account.

8.10.5. MM_GetInvoice

```
text = MM_GetInvoice(invoice, account)
```

Returns an invoice specified by invoice in the specified account.

8.11. Billing Kits

The billing kits provide plug-in billing systems with a very simple API.

8.11.1. MM_GetBillingKitList

```
array = MM_GetBillingKitList()
```

This function returns an array of billing kit names.

8.11.2. MM_GetBillingKit

```
inf = MM_GetBillingKit(kit)
```

This function returns an Inf table for the specified billing kit.

8.11.3. MM_GetBillingKitFields

```
array = MM_GetBillingKitFields(kit)
```

This function returns a list of fields used in the given billing kit. Fields are user-definable values given to the kit.

8.11.4. MM_GetBillingKitSetup

```
inf = MM_GetBillingKitSetup(kit)
```

This function returns an Inf containing the fields and their current values for the given billing kit.

8.11.5. MM_SetBillingKitSetup

```
MM_SetBillingKitSetup(inf, kit)
```

This function stores a (modified) inf table representing the kit fields and values.

8.11.6. MM_PrepareBillingKit

```
text = MM_PrepareBillingKit(kit, account, priceoption,
taxrate)
```

This function prepares an HTML form for the given billing **kit**, **account**, **priceoption** and **taxrate**. If the **taxrate** is -1 then the rate is taken from the account's region.

The returned text is the opening part of the form. To make a complete form you should close the form with your own submit button. For example :—

```
<%= MM_PrepareBillingKit("worldpay", "01234", "std") %>
<input type=submit value="Purchase">
</form>
```

8.11.7. MM_GetInvoiceTemplateList

```
array = MM_GetInvoiceTemplateList()
```

This function returns an array of available invoice templates.

8.12. Bandwidth Charging

These functions manage the excess bandwidth billing system.

8.12.1. MM_GetExcessCharges

```
string = MM_GetExcessCharges(account, formatlevel)
```

This function returns the current excess charges on an account. The format of the returned string is dependant on the **formatlevel**, which can be one of :—

- | | |
|---|--|
| 0 | Just the price is returned as a floating point number |
| 1 | The price including currency is returned |
| 2 | The price including currency and the actual excess amount is returned. |

8.12.2. MM_ClearExcessCharges

```
MM_ClearExcessCharges(account)
```

This function clears the excess charges for the current account. This should be called when excess charges for the previous month have been paid.

8.12.3. MM_IsAccountBandwidthLocked

```
bool = MM_IsAccountBandwidthLocked(account)
```

Returns true if one of the two following conditions have been met :—

- Excess charges are not enabled and bandwidth consumed in the current month has exceeded the account limit
- Excess charges are enabled but the bandwidth limit was exceeded in the previous month and the excess charges are still outstanding

8.12.4. MM_ExpandAccountMacros

```
text = MM_ExpandAccountMacros(account, basetext)
```

This function takes the text given in **basetext** and expands the account macros that appear in that text for the given account. For example, **#CONFIG.PASSWORD#** would be replaced with the password for the given account.

8.13.Regions

These functions manage the regions in MailMaestro.

8.13.1. MM_LoadRegions

```
array = MM_LoadRegions()
```

This function returns an array of region definitions. The functions below should be used to manipulate this data.

8.13.2. MM_StoreRegions

```
MM_StoreRegions(array)
```

This function stores a (modified) region array. Only store arrays returned by `MM_LoadRegions()`.

8.13.3. MM_GetRegionProp

```
obj = MM_GetRegionProp(array, region, prop)
```

This function returned a specific property for a specific region from the region list. For example :—

```
var regions = MM_LoadRegions();
var description_for_USA = MM_GetRegionProp(regions, "USA",
MM_REGION_DESCRIPTION);
```

8.13.4. MM_GetRegionPropIndex

```
obj = MM_GetRegionPropIndex(array, index, prop)
```

This function is similar to `MM_GetRegionProp()` except that instead of specifying the region, the region index is given instead. This should be in the range 0 to `(array.length - 1)`.

8.13.5. MM_SetRegionProp

```
MM_SetRegionProp(array, region, prop, value)
```

This function updates a specific property in the region array.

8.13.6. MM_SetRegionPropIndex

```
MM_SetRegionPropIndex(array, index, prop, value)
```

This function is the same as **MM_SetRegionProp()** except that instead of specifying the region, the region index is given instead.

8.13.7. MM_DeleteRegion

```
MM_DeleteRegion(array, region)
```

This function removes the specified region from the region array.

8.14. Context Sensitive Help

The help system provides a simple method for giving your customers access to on-line documentation.

8.14.1. MM_Help

```
text = MM_Help(item)
```

This function returns an HTML string that is displayed as a question mark in place. When the user clicks on this question mark, the context-help window is opened and the specific item is focused. Item should be a keyword (**A NAME**) somewhere in the chelp.htm file.