Manual for E-mail's Another Rule's Transfer Helper Version 3.5.0

Masaharu FUJITA

平成 15 年 7 月 15 日

1 はじめに

携帯電話では、E-mail を受けられるサービスがあり、自分のアドレスに来た E-mail を携帯電話に転送して、E-mail をリアルタイムにチェックするという使い方をしている方もいると思います。

しかし、携帯電話では、キャリアによって、受信できる文字数の制限が存在したり、文字数の数え方が特殊だったりします。また、単に全ての E-mail を転送するだけでなく、送らなくて良い E-mail や、そのまま送って欲しい E-mail もあるはずです。

以上のようなこと機能を持つスクリプトに大久保正彦さん (`ohkubo@rie.h.kobe-u.ac.jp`) の作られた sky.pl (`http://rie.h.kobe-u.ac.jp/~ohkubo/script.shtml`) というものがあります。

私も最初はこれを使っていましたが、2000 年の年末に齋藤友克さんから

- ヘッダ全体で パターンマッチングして欲しい

という提案がありました。これは、

- Received: の部分で パターンマッチすることによって、ある特定の場所を経由して来たメール毎に転送の設定ができる。

という利点があります。具体的には、直接受け取ったメールに関しては、転送したり、ある domain を通過して来たメールに関しては、転送しなかったり、という設定が可能になります。これは、特に、宛先 (To) や、差出人 (From) が一定でないメーリングリストの転送の設定に有効だと思われます。

また、sky.pl の作者である大久保正彦さんに教えて頂いたのですが、迷惑メール (SPAM) には、ある程度決まった形式があり、X-Mailer: で区別できる場合もあるそうです。

sky.pl には、残念ながら、ヘッダ全体でのパターンマッチング機能がないため、新たにプログラムを作ることにしました。目的は、

- sky.pl+ α で自分の使っている機能を実装する
- ヘッダ 部分の パターンマッチの機能も実装する

ということです。このスクリプトを E-mail's Another Rule's Transfer Helper (この最初の一文字を各々取って来て、earth.rb) と名付けました。名前 (earth.rb) からもわかるように、sky.pl の影響をかなり受けています。(以上 2001 年 3 月 30 日以前)

この様にして、最初の Version が完成しました。しかし、使用しているうちに、

- ヘッダ部分の パターンマッチの機能も全ての ヘッダに対して行うことはできない。
- URL や E-mail address が途中で切れてしまうことがある。

- 電話番号 (らしき番号列) の前には、TEL: をつけて欲しい。
- どのルールにマッチしたかを知りたい。
- ヘッダ情報等を最初のメールで読みたい場合がある。
- Return-Path: を書き換えない場合があるとき、エラーメールが元の送信者に戻る場合がある。(とは言ってもかなり稀な場合ですが....。)

という不満点が出て来ました。これらを修正して、最初から書き直したのが、Version 2 です。

全てのヘッダでのパターンマッチングに関しては、『Ruby を 256 倍使うための本　無道編』の著者の青木峰郎さんが作られている `TMail` (`http://www.loveruby.net/ja/prog/tmail.html`) を使っています。`TMail` は、メールを簡単に扱うためのクラスライブラリです。Version 2 では、これを使うことによって、全てのヘッダでのパターンマッチングを実現させたつもりです。(以上 2001 年 11 月 3 日以前)

Version 2 が完成し、少ししたら、`TMail` が version 1.0 に向けて仕様変更を、頻繁に行うという話を聞きました。(この version が 0.10.X の様です。) 残念ながら、earth.rb は、0.10.X では既に動かないようで、他の方が書かれたライブラリを使う便利さと同時に仕様変更されたときの面倒臭さを感じてしまいました。

そこで、Version 3 では、

- ruby に標準で付属しているクラスライブラリのみを使った実装
- POP および APOP でメールを取得して、転送する機能の実装
- HTML メールへの対応
- SMTP 認証への対応

を行いました。

POP, APOP に対応した理由は、ADSL や CATV 等、接続料定額の環境の方が多くなったため、24 時間稼働の自宅のパソコンからプロバイダのメールをチェックできたら、便利だと考えたためです。

HTML メールへの対応というのは、単にタグを取り除く関数を作って実装しているだけなので、あまり期待しないで下さい。

また、Version 2 でのいくつかのバグも潰れているはずです。もちろん、新たなバグが、できている可能性もあります。(以上 2002 年 5 月 2 日以前)

earth.rb では、以下のような環境が必要です。

- プログラミング言語 ruby がインストールされている。
- 以下のどちらかの環境
  - sendmail または、qmail で E-mail の転送の設定ができる。
  - POP または、APOP でメールを取得でき、 cron 等の定期的なコマンドの設定ができる。

作者の動作確認の環境は、sendmail + ruby-1.6.4 です。開発には、ruby-1.6.6 を使いました。残念ながら、開発・動作確認ともに PC-UNIX 上でのみしか行っていません。(Windows でも動くとは思っています。)

earth.rb でできることは、以下の通りです。

- E-mail を一つのアドレスに転送できる。
- E-mail の pattern ごとに規則を決めて、柔軟に転送の方法を設定できる。
- 代行送信。

- E-mail を転送する時間の設定。
- パターンマッチングでは、ヘッダの全ての行を使うことも可能。

できないこと、わかっていないこと、不満は、以下の通りです。

- バグの摘出が完全でない。特に作者が使ってない
  - J-PHONE 以外のキャリア (会社) への対応
  - POP によるメールの取得
  - SMTP 認証
  - Suspend の動作

  には、かなりのバグが潜んでいると思われます。
- 動作確認も完全でない。
- 現在のところ、J-PHONE のロングメールサービスと E-mail サービスのテンプレートしかできていない。
- sky.pl の方が転送するメールの Byte 数をうまく小さくできている。
- 日本語しか扱えない。
- 一度にたくさんのメールを受信して、分割して送信しようとすると earth.rb が増殖し、サーバに負荷をかけてしまう。
- 作者が RFC 等の E-mail の規格を知らないため、規格通りかどうか分からない。
- src code が汚い (オブジェクト指向らしくない)。
- 作者に英語力がないため、変数の名前、コメントの英語にセンスがない。

ライセンスに関しては、GPL になっているので、自由に使えると思っています。GPL の詳細に関しては、配布物に含まれる gpl.txt か、GNU の WWW site を御覧頂き、それに納得されてから、使用されるようにお願いします。

何か、意見、要望、バグ、その他の修正 (英語の部分も含む) 等がありましたら、`m@fjts.org` まで、御連絡ください。できる限り対応したいと思っています。また、新しい version の earth.rb に関しては、`http://fjts.org/~m/Soft/Ruby/earth.rb/` にて公開したいと思っています。

2 簡単な設定方法

2.1 最低限の設定 (メール転送を使う場合)

1. まず、ダウンロードしたものを展開します。
2. earth.rb と earth.sh, lib ディレクトリ (ディレクトリ内のファイルが必要です) を同じディレクトリに置いてください。
3. このディレクトリに sample 以下の earth_setting.rb.sample.jsky(ロングメール用の設定ファイル) か earth_setting.rb.sample.sky(普通のメール用の設定ファイル) を earth_setting.rb という名前にして置いてください。

4. earth_setting.rb を編集する。

   - `To` には、携帯電話の E-mail address を書いてください。
   - `ReturnPathAddress` には、エラーメールの返る E-mail address を書いてください。この earth.rb を設置する場所の E-mail address がいいと思われます。
   - `SmtpHost` には、SMTP サーバを書いてください。
   - `SmtpPort` には、SMTP の使う port が特殊な場合は、変更してください。(普通は変更しなくても動くはずです。)

この設定の後、 earth.rb のあるディレクトリに移動して、

```
% ruby earth.rb -nospool -nodb < 展開したディレクトリ/testmail
```

を実行してください。このとき、エラーが出ずに以下のような出力がされれば、設定はうまく行っていると思われます。(ただし、E-mail address の設定等によって、実際に出力される内容は異なります)

```
*** Header part ************************************************************
Return-Path: <rp@example.ne.jp>
From: rp@example.ne.jp
To: p@example.ne.jp
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Type: Text/Plain; charset=iso-2022-jp
X-Mailer: earth.rb version 3.4.0

**** Body part **************************************************************
== Infomation ============================================== 30 byte ==========
Subject: 0/m@fjts.org
------------------------------------------------------------------------------
From: m@fjts.org
Subject: 転送のテスト
To: m@fjts.org
Date:
Match: default
------------------------------------------------------------------------------
```

```
*** Header part ************************************************************
Return-Path: <rp@example.ne.jp>
From: rp@example.ne.jp
To: p@example.ne.jp
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Type: Text/Plain; charset=iso-2022-jp
X-Mailer: earth.rb version 3.4.0

**** Body part **************************************************************
== No.0 ================================================== 334 byte ==========
Subject: 0-0/転送のテスト
------------------------------------------------------------------------------
これは、転送のテストです。うまくいくんでしょうか？これがうまくいけば、とりあえずの設定はおし
まいです。◆あとは、earth_setting.rb をいじってください。ただし、earth_setting.rb は、Ｒｕ
ｂｙで書かれており、このプログラムで直接読まれている (require されて
------------------------------------------------------------------------------

== No.1 ================================================== 334 byte ==========
Subject: 0-1/転送のテスト
------------------------------------------------------------------------------
いる）ため、earth_setting.rb に何らかのエラーがあれば、このプログラムは、動きません。◆もし、
設定後に動かなくなった場合は、log/messages や log/wrapperlog を見てください。log/wrapperlog
には、エラーメッセージか書き込まれるようにしてあります
------------------------------------------------------------------------------

== No.2 ================================================== 110 byte ==========
Subject: 0-2/転送のテスト
------------------------------------------------------------------------------
。◆ちなみに、このソフトウェアのライセンスは、ＧＰＬです。
------------------------------------------------------------------------------
```

次に実際に携帯電話に転送できるかどうか、のテストをします。これは、先程のものに -mail というオプションをつけて行います。

```
% ruby earth.rb -nodb -mail < 展開したディレクトリ/testmail
```

これで、先程の出力が、携帯電話に転送されれば、OK です。(1 分くらいプロンプトが戻って来ないことがありますが、無限ループに入っているわけではありません。)

最後にメールの転送の設定です。sendmail の場合は、ホームディレクトリの下の .forward に以下のような記述をします。

```
\ReturnPathAddress で 設 定 し た E-mail address, "| earth.rb が あ る ディレ ク ト
リ/earth.sh -mail"
```

これで自分で出したメールが携帯電話に転送されれば、一番簡単な設定は、終りです。POP を使う場合に関しては、次の節で説明します。

2.2 最低限の設定 (POP を使う場合)

POP でメールを取得する場合には、

1. earth_setting.rb の編集
2. 定期的に earth.sh を実行するための環境設定

が必要です。

まず、earth_setting.rb の中で

```
GetMailType = 'POP'
```

としてください。次に PopSettings を設定します。これはリスト形式で書く必要があり、

0 番目 POP の種類です。'POP' または、'APOP' を書いてください。

1 番目 POP サーバ名を書いてください。

2 番目 POP サーバにアクセスするためのユーザ名を書いてください。

3 番目 POP サーバにアクセスするためのパスワードを書いてください。

4 番目 POP サーバにアクセスするための port number を書いてください。この部分は省略可能で、省略した場合は、110 となります。

の様になっています。例えば、

- server1 に APOP で user1 で passwd1 というパスワードでアクセスする
- server2 に 10110 の port で POP で user2 で passwd2 というパスワードでアクセスする

という場合は、

```
PopSettings = [
  ['APOP', 'server1','user1', 'passwd1'],
  ['POP', 'server2','user2','passwd2', 10110],
]
```

の様に書きます。

次に定期的に earth.sh を実行するための環境設定ですが、ここでは、UNIX で良く使われている定期的にコマンドを実行するデーモン cron についての設定例をあげます。例えば、ユーザ名が user で、earth.sh が/home/user/earth.sh に置いてあって、30 分毎にメールをチェックしに行く場合は、/etc/crontab に

```
*/30    *       *       *       *       user    /home/user/earth.sh
```

というエントリを加えた後、cron を HUP(再起動) して下さい。もちろん、crontab コマンドが使える場合は、そちらで設定してもらってもかまいません。

2.3 E-mail 転送のルールの設定

2.3.1 設定の例

earth.rb では、E-mail ごとに転送する方法を設定できます。例えば、

1. ヘッダに hogehoge の含まれている E-mail は、文字数制限なしで転送する。
2. From アドレス が、 e@example.net の E-mail は転送しない。
3. example.com を通過したメールは、転送しない。(つまり、Received に example.com が含まれている場合。)
4. Subject の先頭が、aho の場合は、335 Byte × 7 通で転送する。

です。

earth.rb では、このようになルールを設定するために earth_setting.rb の

```
Rule = [
  ['subject', '^Returned mail:', 0, 0],
  ['from', 'MAILER-DAEMON@%email\.sky\.(cdp|kdp|tdp|dtg)\.ne\.jp)?@email\.sky\
.(cdp|kdp|tdp|dtg)\.ne\.jp', 0, 0],
  ['from', 'MAILER-DAEMON@sky\.(tu-ka|tkk|tkc)\.ne\.jp', 0, 0],
  ['from', 'MAILER-DAEMON@jp-[dhrtcknsq]\.ne\.jp', 0, 0],
  ['from', 'MAILER-DAEMON@docomo\.ne\.jp', 0, 0],
  ['from', 'MAILER-DAEMON', 0, 0],
  ['default', nil],
]
```

に手を入れます。先の 1., 2., 3. のような設定をしたい場合には、

```
  ['all', 'hogehoge', -1, -1],
  ['from', 'e@example\.net', 0, 0],
  ['received', 'example\.com', 0, 0],
  ['subject', '^aho', 335, 7],
```

を Rule の中に書き入れます。

2.3.2 Rule の形式

次に、この `Rule` の書き方について説明します。

0 番目 ヘッダ で パターンマッチする際にどんなパターンで行うか？というタイプです。ヘッダのエントリ ('to', 'from', 'subject', 'received' 'message-id' 等) の他に以下のようなタイプを指定できます。

- 'all': ヘッダ の全ての行で パターンマッチを行う。
- 'tocc': To, Cc のアドレスによって、パターンマッチを行う。
- 'destinations': 'tocc' と同じ。
- 'default': Default の動作。

1 番目 パターンマッチする際のキーワードを書き込みます。Regexp.new(キーワード) として、正規表現を作って、判定するので、この部分は、 Ruby の正規表現を使うことができます。

2 番目 メールを転送する際の最大の Byte 数を書き入れます。この部分は省略可能で、省略した場合は、`DefaultCutMaxbyte` の値が適用されます。

3 番目 メールを転送する際に 2 番目の Byte 数でメールを分割する訳ですが、それを何通送るのか、という設定です。-1 は無限大を表します。この部分も省略可能で、省略した場合は、`DefaultMaxNumber` の値が適用されます。

4 番目 メールを再転送する場合の、メールを転送する際の最大の Byte 数を書き入れます。再転送に関しては、後程説明します。この部分は省略可能で、省略した場合は、`DefaultRegetMaxbyte` の値が適用されます。

5 番目 メールを再転送する場合に、それを何通送るのか、という設定です。3 番目と同様に -1 は無限大を表します。更に、この部分も省略可能で、省略した場合は、`DefaultRegetMaxNumber` の値が適用されます。

6 番目 隠しコマンドです:-)。 この動作について知りたい方は、申し訳ありませんが、src code をお読みください (e.g. `grep data[6] earth_lib.rb`)。よく分からない場合は、使わない方がいいと思います。簡単な例としては、以下のようなものが使えます。

- "$just = true": メールを分割せずに、そのまま転送します。
- "$noinfo = true": ヘッダ情報を転送しないようにします。
- "$nobody = true": メールの本体を転送しないようにします。

この `Rule` ですが、最初の要素から パターンマッチを行います。つまり、

- **上に書いたルール程、優先順位が強くなります。**

3 earth_setting.rb の設定について

ここでは、目的別の earth_setting.rb の設定方法について述べて行きます。ただし、earth_setting.rb は、Ruby で書かれており、このプログラムで直接読まれている (require されている) ため、earth_setting.rb に何らかのエラーがあれば、このプログラムは、動きません。

もし、設定後に動かなくなった場合は、log/messages や log/wrapperlog を見てください。log/wrapperlog には、エラーメッセージか書き込まれるようになっています。

3.1 分割して送るメールの Subject に対する設定

分割して送るメールに Subject をつける場合は、

```
UseSubject = true
```

としてください。更に以下の 2 つの設定が可能です。

- `SubjectFormat`: Subject の形式を決める。この部分は、eval で展開しているため、" でくくった、Ruby の文を書き込めます。
  以下が使えそうな変数です。

  - `@mail_number`: 保存されているメールの番号 (0 から始まります)
  - `@count_number`: 何番分割目のメールであるのか？(0 から始まります)
  - `@subject`: そのメールの Subject
  - `@from`: そのメールの From の E-mail address
- `SubjectMaxbyte`: 分割されたメールにつける Subject の最大の Byte 数

3.2 分割したメールの最初につける形式

分割した最初に様々な情報を付け加えることによって、メールが見やすくなったり、重要な情報が得られたりします。

- `FirstBodyHead`: 最初のメールの冒頭につく形式。
- `NextBodyHead`: 2 通目以降のメールの冒頭につく形式。

この部分は、`SubjectFormat` と同じように eval で評価されているため、Ruby の文を書くことができます。使えそうな変数に関しても `SubjectFormat` を御覧下さい。また、`nil` と設定することによって、省略することもできます。

3.3 ヘッダ情報

earth.rb では、送られて来たメールのヘッダ情報の一部を加工して、転送するという機能もあります。ただし、このヘッダ情報に関しては、メールの大きさを計算していないため、最大分割バイトを越える場合もあります。

この機能を使いたくない場合は、

```
SendHeaderInfo = false
```

とするか、プログラムを実行する際の引数に

```
-noinfo
```

をつけてください。

さらに次のような値を設定することができます。

- `InfoMailSubject`: ヘッダ情報を転送するメールの Subject の形式。`SubjectFormat` 同様、eval によって評価されるので、Ruby で書くことができます。意味がありそうな変数として、`@mail_number`, `@subject`, `@from`, の他に
  - `@tocc`: To と Cc のメールアドレス
  - `@match`: どの Rule にマッチングしたのか？

  があげられます。
- `InfoMailBody`: ヘッダ情報を転送するメールの本文の形式。`SubjectFormat` 同様、eval によって評価されるので、Ruby で書くことができます。意味がありそうな変数として、`@mail_number`, `@subject`, `@from`, `@tocc`, `@match` の他に

  - `@date`: メールヘッダに書かれている日時

  があげられます。

- `InfoMailFromAddress`: ヘッダ情報を転送するときに使う差出人のメールアドレス。
- `MaxInfoAddressNum`: ヘッダ情報を転送するときに宛先 (To と Cc) の E-mail address をつける個数。
- `HeaderInfoLastSend`: ヘッダ情報を最後に送る場合は、true を設定して下さい。最初に送る場合は、false で、これがデフォルトとなっています。

3.4 改行に対する設定

改行が 2 回以上続いた場合に置き換えたい場合、

```
ChangeLineFeed = true
```

としてください。置き換える文字は、

```
LineFeedCharacter
```

で設定できます。

3.5 Spool されるメールに関する設定

earth.rb では、転送したメールを一旦保存しています。これによって、後程、説明する、メールの再転送および、転送する時間の設定が、可能になっています。もし、メールを保存したくない場合は、

```
UseSpool = false
```

とするか、プログラムを実行する際の引数に

```
-nospool
```

をつけてください。また、保存するメールの最大個数ですが、

```
SpoolNumberMax
```

で設定できます。また、メールは、`spool` ディレクトリに番号で保存されるようになっています。

3.6 カットする行の設定

無条件にカットする行を設定できます。この設定は、

```
CutLinePattern
```

によって行ってください。(ここでも Ruby の正規表現が使用できます。) ただし、これは、必要な部分も切り取ってしまう可能性がありますので、注意して使ってください。

もし、この機能を使いたくない場合は、

```
CutLineFlag = false
```

としてください。

3.7 データベースの設定

earth.rb では、過去に来たメールを用いて、2 行以上重複する行 (改行のみの行は除く) をカットする機能があります。これにより、長い signature がカットできる場合もあります。まず、過去のメールを From アドレスごとにその E-mail address の名前をファイル名として db ディレクトリ以下に保存されます。そして次に、差出人が、保存されているファイル名のアドレスでであるメールが来た場合には、今まで来たメール (保存されている内容) と比較して、2 行以上重複する行をカットする、という機能です。この機能を使いたくない場合は、

```
UseDB = false
```

とするか、プログラムを実行する際の引数に

```
-nodb
```

をつけてください。また、db ディレクトリ以下のファイルの大きさの上限は、

```
DBMaxFilebyte
```

(単位は Byte) によって設定できます。

3.8 連続する文字を一文字に圧縮する

連続する文字列を一文字に圧縮する機能もあります。これを使うためには、

```
UseSqueeze = true
```

としてください。
実際に圧縮する文字に関しては、

```
SqueezeCharacter
```

で設定できます。

3.9 返信の行に対する設定

返信の際につく記号「>」、「|」または、「:」の行を全て消したい場合には、

```
EditReplyLine = true
WriteOneReplyLine = false
```

としてください。一行だけ残す場合には、

```
EditReplyLine = true
WriteOneReplyLine = true
```

としてください。また、全て残す場合は、以下のようにします。

```
EditReplyLine = true
```

3.10 SMTP 認証の設定

earth.rb は、SMTP 認証 にも対応しています。この機能を有効にしたい場合は、

```
UseSmtpAuth = true
```

としてください。

認証形式は、現在、Ruby がサポートしている CRAM_MD5 と PLAIN のみです。(動作確認は、CRAM_MD5 のみしか行っていません。) これらを踏まえて、以下の変数も設定してください。

- `SmtpAuthType`: SMTP 認証で使う認証形式です。現在使えるのは、Ruby がサポートしている
  - `'cram_md5'`
  - `'plain'`

  のみですが、将来的には増えるかも知れません。
- `SmtpUser`: SMTP 認証で使う際のユーザ名を代入してください。
- `SmtpPasswd`: SMTP 認証で使う際のパスワードを代入してください。

3.11 キャリアごとに依存する設定

まず、作者は J-PHONE のみしか携帯電話を持っていないため、J-PHONE の話が中心になることをお断り致します。

J-PHONE の携帯電話の大部分は、切替えコードの挿入により、全角文字と半角文字の境界で Byte 数が余分にカウントされます。このため、半角英数を使うよりも全角英数を使った方が多く文字数を送ることが可能な場合があります。この様な場合 (J-PHONE) を多少考慮した Byte 数のカウント、半角英数と全角英数の変換を行う場合には、

```
PhoneType = 'J-PHONE'
```

として下さい。また、Byte 数のみをカウントする

```
PhoneType = 'J-PHONE_CUTONLY'
```

というタイプも存在します。もし、違うキャリアで、余分にカウントされない場合には、`'J-PHONE'` 以外の文字列を設定して下さい。別関数を使って、Byte 数をカウントするようになっています。ただし、この別関数に関するデバックは十分には行われていませんので、ご注意下さい。

`MailbyteContent` は、メールを送る際、本文以外で、メール全体の Byte に影響を与える要素を書いていきます。例えば、ロングメールでは、

- Subject の Byte 数。
- From アドレスの Byte 数。

が、関係するので、次のように書いてあります。

```
MailbyteContent = "subject_byte + from_byte + 2"
```

ここで、2 は、Magic number です。:-)

3.12 コントロールコマンド

earth.rb には、

- メールの再転送
- メールの送信代行

という機能があります。携帯電話からある形式のメールを送信することによって、この機能を使うことができます。(ある形式の文字列-コマンドを送ることによって、機能を制御できるため、「コントロールコマンド」としました。) この機能を使いたい場合は、

```
UseCtrlCmd = true
```

としてください。

しかし、これは、ヘッダ の偽造によって他人からのメールでもできる可能性があるので、注意が必要です。これを防ぐために以下のような定数を設定可能にしました。

- `CtrlCmdPassword`: パスワード。E-mail は、平文で流れるので、あまり意味がないかもしれませんが、ないよりはマシだと思い、作りました。E-mail で使えるどんな文字列でも可能です。ただし、Default の値は必ず、変更してください。
- `CtrlCmdReceived`: Received: 部分の pattern 設定。
- `CtrlCmdMessageID`: Message-Id: 部分の pattern 設定。
- `CtrlCmdFromList`: コントロールコマンドを受け付ける From E-mail address のリスト。これによって、特定の E-mail address からのメールのみからの要求にしか答えないようにできます。
- `CtrlCmdTo`: コントロールコマンドのメールの To の E-mail address を設定。E-mail address が 2 つ以上設定できる場合に、他の人から受け取る E-mail address とコントロールコマンドを実行するために送る E-mail address を区別して使うために設定できるようにしました。

また、その他の設定については、以下の通りです。

- `VicariousFrom`: 送信代行の際 From: につく E-mail address。
- `VicariousSubject`: 送信代行の際の Subject。
- `VicariousBcc`: 送信代行するときに加える E-mail address。例えば、`To` で指定した自分の携帯電話と E-mail address ¡example@example.org¿へメールを送信したい場合は、

  ```
  VicariousBcc = [To, 'example@example.org']
  ```

  のように書いてください。
- `VicariousSignature`: 送信代行の際につく signature。
- `SubjectPause` メールを送る際に件名を設定する時の区切りの文字。

3.12.1 再転送

メールの再転送とは、一旦転送されたメールをもう一度携帯電話に転送するという機能です。
再転送をする場合は、

```
UseSpool = true
```

という設定が必要です。
携帯電話から出すメールの形式は、

```
:REGET␣パスワード␣再転送するメールの保存番号,再転送するメールの保存番号,再転送するメールの保存番号,.........
```

を一行で書いて送ってください。例えば、パスワードが、「ほげほげ」で 100 番と 130 番のメールを再転送したい場合は、

```
:REGET␣ほげほげ␣100,130
```

とて、携帯電話から、earth.rb を設置している E-mail address にメールを送ります。
E-mail をそのまま転送して欲しい場合は、

```
:REGET␣ほげほげ␣100␣just
```

とするとできます。

3.12.2 送信代行

メールの代行送信とは、携帯電話の E-mail address ではなく、携帯電話から、設定した E-mail address を From として出すという機能です。携帯電話の E-mail address を知られたくない相手にメールを出す場合には、便利な機能だと思われます。
送信代行には、

- 来たメールに対する返信。
- あるメールアドレスにメールを送る。
- あるメールアドレスに件名を設定してメールを送る。

という 2 つがあります。

来たメールに対する返信

```
:REPLY␣パスワード␣保存されているメールの番号␣内容
```

のように書きます。Subject ですが、「Re: 返信するメールの Subject」となります。余分な Re: や RE: は、外すようにしてあります。例えば、メール番号 50 に返事を書きたい場合は、

```
REPLY␣ほげほげ␣50␣代行送信の返信のテストです。
ちゃんと届くかな？
```

のように書きます。

あるメールアドレスにメールを送る

```
:SEND␣パスワード␣送信するメールアドレス,送信するメールアドレス,....␣内容
```

のように書きます。例えば、

```
:SEND␣ほげほげ␣user@example.ne.jp␣代行送信のテストです。
ちゃんと届くかな？
```

のように書きます。件名 (Subject) は、VicariousSubject の値になります。

あるメールアドレスに件名を設定してメールを送る

```
:SEND␣パスワード␣送信するメールアドレス,送信するメールアドレス,....␣件名＊内容
```

のように書きます。ただし、”＊” は、SubjectPause の値です。例えば、テストという件名をつけたい場合、

```
:SEND␣ほげほげ␣user@example.ne.jp␣テスト＊件名設定の代行送信のテストです。
ちゃんと届くかな？
```

のように書きます。

3.13 メールを転送しない時間の設定

メールを携帯電話に転送しない時間も設定することができます。これを有効にするには、

```
UseSuspend = true
```

としてください。

もし、メールを転送しない時間が過ぎてから来た最初のメールを受け取ることによって、転送しない時間に受け取ったメールを一斉に転送します。

設定できる項目は以下の通りです。

- 平日の転送しない時間の設定
- 休日の転送しない時間の設定
- 休日の設定
- 転送しない間でも送る E-mail のパターンの設定

3.13.1 平日の転送しない時間の設定

WeekdaySuspendTime で設定します。例えば、夜中の 23:30 から 朝の 8:00 までメールを転送しなくて良い場合には、

```
WeekdaySuspendTime = [
  [ 2330, 2359 ],
  [ 0, 800 ],
]
```

WeekdaySuspendTime の配列の要素ですが、必ず、左の数よりも右の数の方が大きくなるように書いてください。

3.13.2 休日の転送しない時間の設定

HolidaySuspendTime で設定します。WeekdaySuspendTime とほぼ同じですが、これは、休日 (Holiday) で設定された日に適用されます。

3.13.3 休日の設定

Holiday で設定します。Default では、土曜日、日曜日、1/1, 1/2, 2/11, 4/29, 5/3, 5/4, 5/5, 7/20, 9/15, 11/3, 11/23, 12/23, 12/30, 12/31 が設定されています。この設定は、以下の通りです。

```
Holiday = [
  [ "%a", '(Sun|Sat)' ],
  [ "%m%d", '010[12]' ],
  [ "%m%d", '0211' ],
  [ "%m%d", '0429' ],
  [ "%m%d", '050[345]' ],
  [ "%m%d", '0720' ],
  [ "%m%d", '0915' ],
  [ "%m%d", '11[02]3' ],
  [ "%m%d", '12(23|30|31)' ],
]
```

まず、0 番目は、どんな形式で日時を判定するか、で Time の strftime の形式が取られます。これは、Ruby のマニュアルで確認してください。1 番目は、実際に判定される日時の pattern です。これは、Ruby の正規表現が使えます。

3.13.4 転送しない間でも送る E-mail のパターンの設定

IgnoreSuspendRule で設定します。この書き方は、前述の Rule とほぼ同じで、

0 番目 ヘッダ で パターンマッチする際にどんなパターンで行うタイプ。

1 番目 パターンマッチする際のキーワード。

となっています。

3.14 その他の設定

3.14.1 MaxbyteOfOneMail

受け取るメールの限界の大きさ (Byte)。

3.14.2 FromType

携帯電話に転送される際の From の E-mail address をどうするか、という設定です。`nil` の場合は、送られて来た E-mail の From アドレスを差出人としてます。差出人を固定したい場合には、その固定したい E-mail address を設定して下さい。

3.14.3 LatencyTime

分割されたメールを送る際に、メールとメールの間に何秒の間隔を開けるのかという設定です。

3.14.4 DefaultCutMaxbyte

メールを転送する際の Default の最大の Byte 数。

3.14.5 DefaultMaxNumber

分割したメールを何通送るのか、という Default の設定。-1 の場合は無限大。

3.14.6 DefaultRegetMaxbyte

メールを再転送する場合の、メールを転送する際の Default の最大の Byte 数。

3.14.7 DefaultRegetMaxNumber

メールを再転送する場合に、それを何通送るのか、という Default の設定。-1 の場合は無限大。

3.14.8 SendNoAtMarkAddress

@(アットマーク) のついてないメールアドレスから来たメールを転送するかどうか。Default の設定は true で @ がなくても転送します。

4 Log について

earth.rb と earth.sh によって、log/messages と log/wrapperlog に log が残ります。それぞれ

- log/messages: earth.rb の動作に関する Log
- log/wrapperlog: earth.sh の実行時エラーに関する Log

という役割を果たしています。

また、log/messages は、ある一定の大きさになると log/messages.bak として保存されます。保存されるのは、一つ前の log/messages のみです。

5 最後に

earth.rb は、メールシステム＆管理者へのエラーメールを出さないように earth.sh の最後に `exit 0` と書いています。このため、このプログラムは必ず正常に終了するはずです。

何か、意見、要望、バグ、その他の修正 (英語の部分も含む) 等がありましたら、`m@fjts.org` まで、御連絡ください。できる限り対応したいと思っています。

Masaharu FUJITA <m@fjts.org>, URL of earth.rb: http://fjts.org/~m/Soft/Ruby/earth.rb/

索 引