Mail System for NORTi

User's Guide

2008 年 9 月版

MiSPO
株式会社ミスポ

2008 年 9 月版で改訂された項目

ページ	更新内容
3	1.4 ファイル構成に暗号処理とサンプルプログラムの記述を追加

2008 年 4 月版(2)で改訂された項目

ページ	更新内容
6	(注 1)に対応認証方式の記述を追加(AUTH CRAM-MD5)

2008 年 4 月版で改訂された項目

ページ	更新内容
全般	語句、言い回しの修正、インデントの修正
5	esmtp_ini を追加
6	SMTP 認証使用時について注 1 に記述
8	esmtp_ini に関する説明を追加
9	EHLO コマンド追加
11,12	smtp_set_opt に、SMTP_SET_PORTNO,SMTP_SETUSER,SMTP_SET_PASS, SMTP_SET_AUTH を追加
奥付	会社住所修正

2005 年 4 月版で改訂された項目

ページ	更新内容
4	1.5 FileSystem を使用する場合を追加

目　次

第１章　導入

1.1 はじめに

MailSystem NORTi は(以降 MailSystem)は NORTi TCP/IP プロトコルスタック) の
アプリケーションレイヤで次の機能を提供します。

・SMTP クライアント

・POP3 クライアント

・MIME サポート機能

本書では MailSystem の使用法についての説明を行っています。
OS やプロトコルスタックの使用方法については各ユーザーズガイドを参照してください。

1.2 特長

・リソースは、ファイルベース、メモリベースのどちらも使用可能です。
・プロトコルスタックの全ソースコードが付属しています（評価版を除く)。
　開発製品毎のライセンス制をとっており、組み込みロイヤリティが無料です。

1.3 制限事項

・μITRON 仕様 OS NORTi の下で動作するよう特化しているため、他の OS へ移植することは
　困難です。
・本システムはクライアント機能のみをサポートします。
・ファイルベースでの動作を行うために NORTi FileSystem またはプロトコルスタックの
　サンプル nonfile.c が使用できます。
・本書に記載されている内容は、予告無く変更されることがあります。

1.4 ファイル構成

NORTi/MAIL/SRC

nonsmtp.c	SMTP プロトコルスタック
nonpop3.c	POP3 プロトコルスタック
nonmime.c	MIME サービス関数
md5.c	暗号処理関数

NORTi/MAIL/INC

nonmail.c	MailSystem ヘッダ
nonmcfg.h	MailSystem コンフィグレーションヘッダ
md5.h	暗号処理関数ヘッダ

NORTi/SMP

ボード固有のサンプルファイル

milxxxx.c	MailSystem ボード固有サンプルプログラムメイン
mailcmd.c	MailSystem サンプルコマンド処理プログラム
mailsmp.h	MailSystem サンプルプログラムヘッダファイル
mailsmp.c	MailSystem サンプルプログラム
mailcfg.c	MailSystem サンプル用コンフィグレーションファイル

1.5 FileSystem を使用する場合

FileSystem を使用する場合はアプリケーションプログラムと nonmime.c コンパイル時に、使用する FileSystem のバージョンに合わせて以下のマクロを定義してください。

マクロ名	
なし	FileSystem を使用しない
NOFILE_VER=1	NORTi 添付の FileSystem(NETSMP¥SRC)
NOFILE_VER=4	NORTi FileSystem Version4

第２章　関数一覧

SMTP クライアント（送信系）

smtp_ini	SMTP の初期化
esmtp_ini	ESMTP の初期化(SMTP 認証使用時)
smtp_set_opt	SMTP のオプション設定
smtp_command	SMTP コマンド処理
smtp_snd_dat	データ送信

POP3 クライアント（受信系）

pop3_ini	POP3 の初期化
pop3_set_opt	POP3 のオプション設定
pop3_command	POP3 コマンドの処理
pop3_rcv_reply	リプライの受信
pop3_rcv_dat	データ受信

MIME サポート機能（送信系）

mime_mak_mail	メールデータの作成

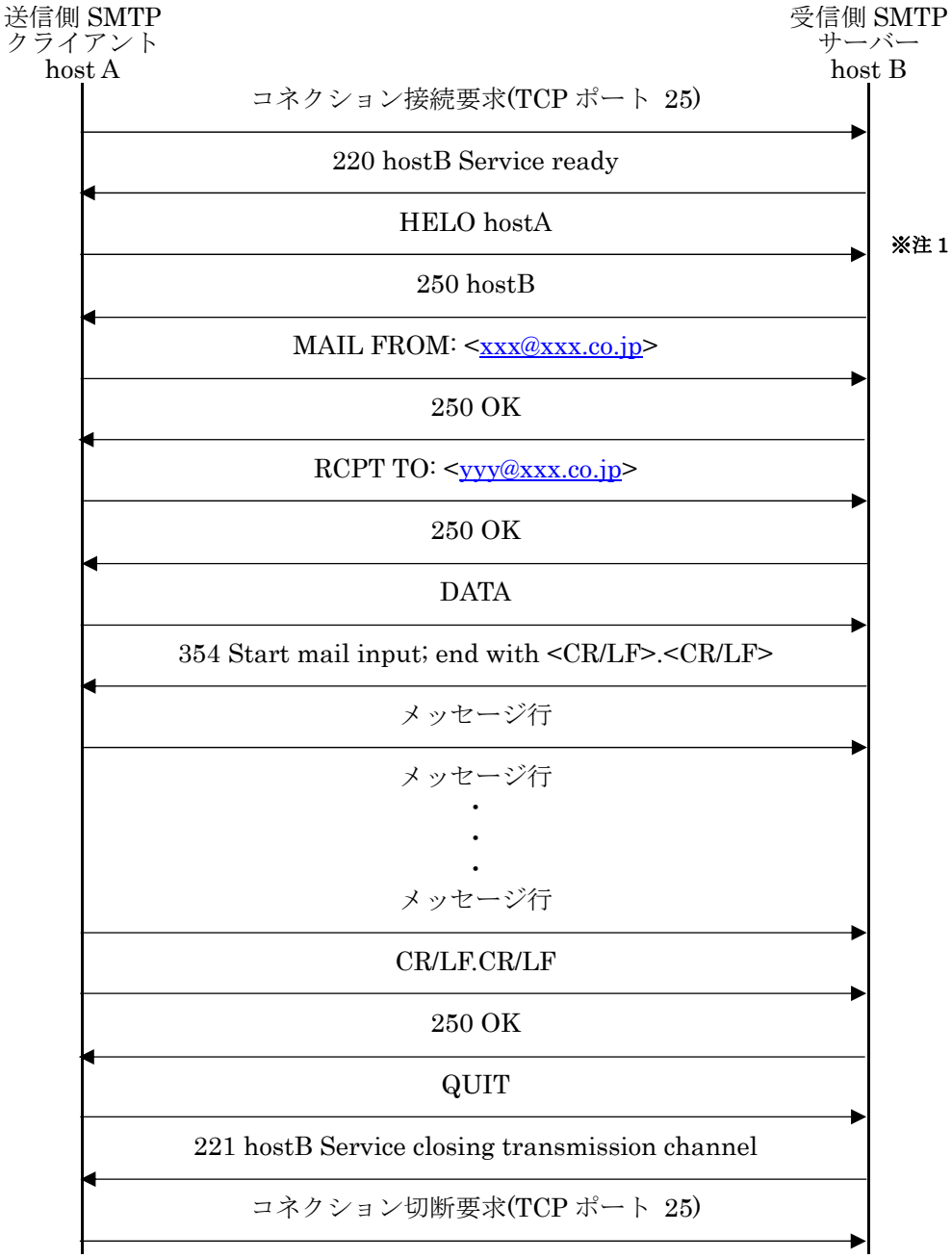
MIME サポート機能（受信系）

mime_get_hinf	受信メールデータのヘッダ情報の取得
mime_get_afname	添付ファイル情報の取得
mime_sep_mail	受信メールデータを分割する

第３章　SMTP クライアント機能

SMTP(Simple Mail Transfer Protocol)プロトコルスタックはメールサーバーへ簡易メール転送を実現します。
本クライアントではメールデータ、メールヘッダの中身には関知しません。

【SMTP の通信例】



※注 1

SMTP 認証使用時は、通信はサブミッションポートに対して行い、コマンドも EHLO を使用します。

EHLO に対するレスポンスコードを自動で判定し、使用可能な認証方式の中から選んで認証を行います。

現バージョンでは AUTH CRAM-MD5,AUTH LOGIN,AUTH PLAIN に対応しています。サブミッションポートを使用するものの、

AUTH レスポンスを返さないサーバに関しては、そのまま認証を行わずサブミッションポートを使った通信となります。

smtp_ini

[機能] SMTP プロトコルスタックの初期化

[形式] ER smtp_ini(T_SMTP *smtp, ID cepid, UW server, SMTP_CALLBACK callback);

smtp	SMTP コントロールブロックのポインタ
cepid	TCP 通信端点 ID(0 指定時は自動設定)
server	SMTP サーバーIP アドレス
callback	コールバックルーチン(現在は未サポートのため NULL を設定)

[戻値]

E_OK	正常終了
E_ID	不正 ID 番号
E_OBJ	TCP 通信端点が生成済み

[解説] SMTP プロトコルスタックの初期化を行います。

[例]
```
T_SMTP smtp; /* SMTP コントロールブロック */
ap_init()
{
    ER ercd;
        ・
        ・
    ercd = smtp_ini(&smtp, 0, smtp_server, NULL);
        ・
        ・
}
```

esmtp_ini

[機能] SMTP プロトコルスタックの初期化(SMTP 認証用 ESMTP モードで使用する場合)

[形式] ER esmtp_ini(T_SMTP *smtp, ID cepid, UW server, SMTP_CALLBACK callback, UH portno ,

char *str_username, char *str_passwd);

smtp	SMTP コントロールブロックのポインタ
cepid	TCP 通信端点 ID(0 指定時は自動設定)
server	SMTP サーバーIP アドレス
callback	コールバックルーチン レスポンスをユーザが解析したい場合に、使用します。 特に指定の必要がない場合は NULL を指定します。
portno	SMTP 認証用ポート番号
str_username	SMTP 認証用アカウント(ユーザ名)文字列へのポインタ
str_passwd	SMTP 認証用パスワード文字列へのポインタ

[戻値]

E_OK	正常終了
E_ID	不正 ID 番号
E_OBJ	TCP 通信端点が生成済み

[解説] SMTP プロトコルスタックの初期化を行います。

[例]

```
T_SMTP smtp; /* SMTP コントロールブロック */
#define ESMTP_PORT 587

void ap_init(void)
{
    ER ercd;
        ・
        ・
    ercd = esmtp_ini(&smtp, 0, smtp_server, NULL, ESMTP_PORT , “tarou”, “hanako123″);
           /* port=587,アカウント=tarou,パスワード=hanako123        */
        ・
        ・
}
```

smtp_command

[機能] SMTP コマンド処理

[形式] ER smtp_command(T_SMTP *smtp, char* command);

smtp	SMTP コントロールブロックのポインタ
command	コマンド文字列

[戻値]

E_OK	正常終了
E_PAR	不正コマンド
E_OBJ	SMTP シーケンスエラー
E_NOEXS	SMTP が未初期化
E_TMOUT	通信エラー
E_CLS	サーバーからの切断要求

[解説] SMTP コマンドを実行します。

コマンド名	パラメータ	内容
HELO	送信側ホスト名	通信路の使用開始宣言
EHLO	送信側ホスト名	通信路の使用開始宣言(SMTP 認証用)
MAIL	送信者名	メールボックス宛のメール送信開始
RCPT	宛先ユーザー名	メール受信者の指定
DATA	メッセージデータ	メール本文の送信開始
NOOP	なし	サーバーが動作しているか確認
QUIT	なし	SMTP の接続を終了
¥r¥n.¥r¥n	なし	データ送信の終了

[例] 一連のシーケンス例

```
send_mail()
{
    ER ercd;
        ・
        ・
    /* EHLO コマンドの送信 */
    strcpy(command, "EHLO MyDomain");
    ercd = smtp_command(&smtp, command);   /* この中で認証まで行います */

    /* 発信元のメールアドレスを送信 */
    strcpy(command, "MAIL <test0@xxx.xxx>");
    print(command);
    print("¥r¥n");
    ercd = smtp_command(&smtp, command);

    /* 宛先のメールアドレスを送信 */
    strcpy(command, "RCPT <test1@xxx.xxx>");
    ercd = smtp_command(&smtp, command);

    /* メールデータの送信 */
    ercd = smtp_command(&smtp, "DATA");
    ercd = smtp_snd_dat(&smtp, mbody, size);
    ercd = smtp_command(&smtp, ".");
ERR:
    /* メール送信完了 */
    ercd = smtp_command(&smtp, "QUIT");
}
```

smtp_snd_dat

[機能] メールデータの送信

[形式] ER smtp_snd_dat(T_SMTP *smtp, VP buf, int size);

smtp	SMTP コントロールブロックのポインタ
buf	送信データへのポインタ
size	送信データの長さ

[戻値]

正の値	送信されたデータサイズ
E_OBJ	SMTP シーケンスエラー
E_NOEXS	SMTP が未初期化
E_TMOUT	通信エラー
E_CLS	サーバーからの切断要求

[解説] メールデータを送信します。

smtp_set_opt

[機能] SMTP オプションの変更

[形式] ER smtp_set_opt(T_SMTP *smtp, INT optname, const VP optval, INT optlen);

smtp	SMTP コントロールブロックのポインタ
optname	オプション名
optval	オプション
optlen	オプションの長さ

[戻値] E_OK 正常終了
E_PAR パラメータエラー

[解説] SMTP のオプションを設定します。

オプション名 optname には次の値を指定できます。

SMTP_SET_IPADDR

SMTP サーバーの IP アドレスを変更します

optval UW 型(4byte)の IP アドレスが格納されたポインタ

optlen 4

例) smtp_set_opt(&smtp, SMTP_SET_IPADDR, &smtp_ipaddr, 4);

SMTP_SET_NIF_BYNAME

SMTP が使用するネットワーク I/F をネットワーク I/F 名から設定します

optval ネットワーク I/F 名

optlen 未使用(0 指定)

例) smtp_set_opt(&smtp, SMTP_SET_NIF_BYNAME, "ppp", 0);

SMTP_SET_NIF_BYCH

SMTP が使用するネットワーク I/F をチャンネル番号から設定します

optval チャンネル番号

optlen 未使用(0 指定)

例) smtp_set_opt(&smtp, SMTP_SET_NIF_BYCH, 1, 0);

※SMTP_SET_NIF_BYNAME および SMTP_SET_NIF_BYCH はマルチチャネル版プロトコルスタック使用時のみ指定可能です。

SMTP_SET_PORTNO

SMTP 認証用サブミッションポートの再設定を行います。

optval UH 型(2 バイト)のポート番号が格納されたポインタ

optlen 2

例) smtp_set_opt(&smtp, SMTP_SET_PORTNO, &port_no, sizeof(port_no));

SMTP_SET_USER

SMTP 認証用アカウント名の再設定を行います。

optval　　アカウント名文字列が格納されたポインタ(char *)

optlen　　31 以下(デフォルト)

例) smtp_set_opt(&smtp, SMTP_SET_USER, user, strlen(user));

SMTP_SET_PASS

SMTP 認証用パスワード文字列の再設定を行います。

optval　　パスワード文字列が格納されたポインタ(char *)

optlen　　31 以下(デフォルト)

例) smtp_set_opt(&smtp, SMTP_SET_PASS, passwd, strlen(passwd));

SMTP_SET_AUTH

SMTP 認証を行うかどうかの再設定を行います。

optval　　UH 型(2 バイト)の TRUE/FALSE が格納された変数へのポインタ

　　TRUE :SMTP 認証を行う。

　　FALSE:SMTP 認証を行わない

optlen　　2

例) smtp_set_opt(&smtp, SMTP_SET_AUTH, do_auth, sizeof(do_auth));

第４章　POP3 プロトコルスタック機能

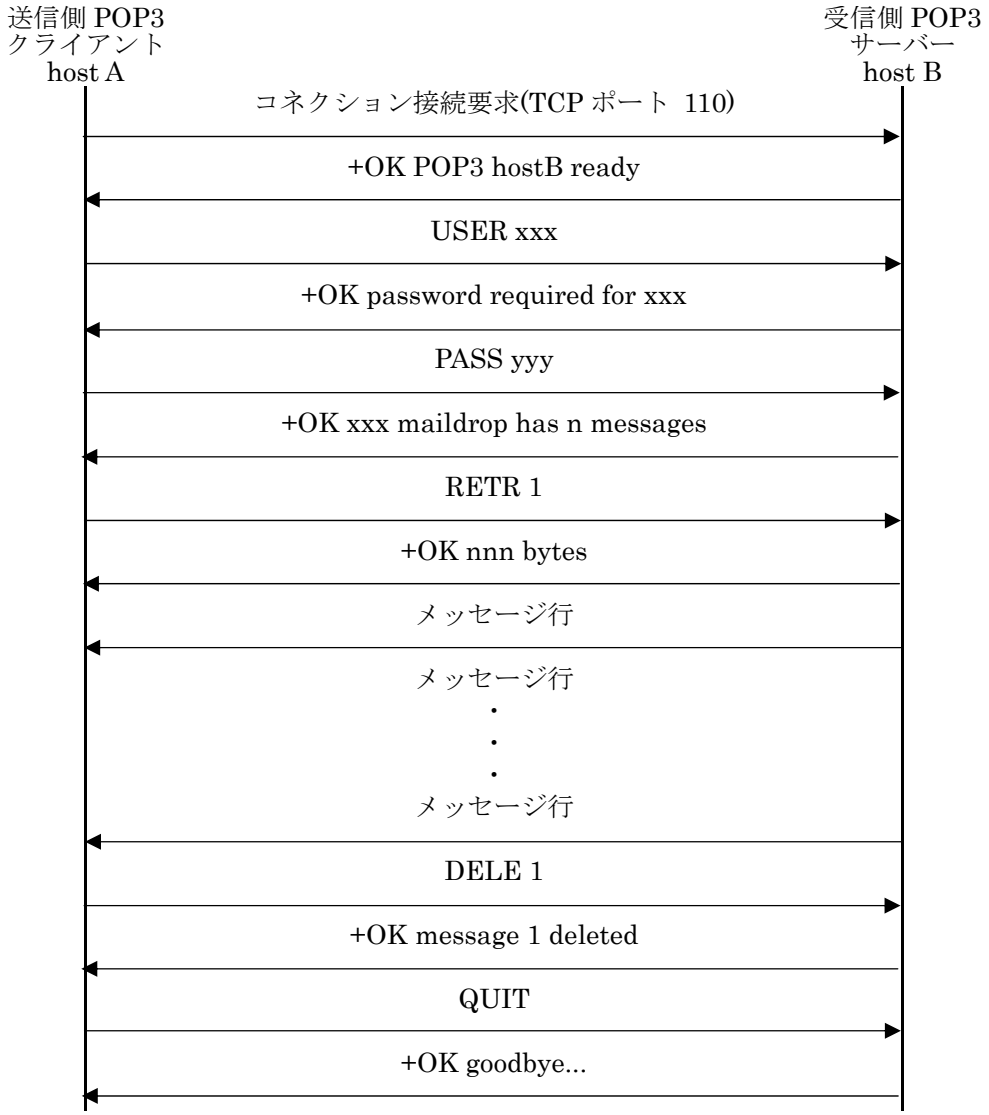
POP3(Post Office Protocol version 3)プロトコルスタックは

・ユーザー認証
・メールボックスからのメールデータの取得
・メールボックスにある全体のメールデータサイズの取得
・個々のメール情報(サイズなど)を取得
・個々のメールヘッダの取得
・メールボックスにあるメールの削除

などを行うことが出来ます。

本プロトコルスタックではメールデータ、メールヘッダの中身には関知しません。

【POP3 の通信例】



pop3_ini

[機能] POP3 プロトコルスタックの初期化

[形式] ER pop3_ini(T_POP3 *pop3, ID cepid, UW server, POP3_CALLBACK callback);

pop3	POP3 コントロールブロックのポインタ
cepid	TCP 通信端点 ID(0 指定時は自動設定)
server	POP3 サーバーIP アドレス
callback	コールバックルーチン(未使用時は NULL を指定)

[戻値]

E_OK	正常終了
E_ID	不正 ID 番号
E_OBJ	TCP 通信端点が生成済み

[解説] POP3 プロトコルスタックの初期化を行い、メールサーバーへ接続します。
コールバックルーチンはメールの受信の時に、規定のバイト数が受信されるたびに呼び出されます。
現在の受信状況をアプリケーションが知りたい場合に便利です。

[例]
```
T_POP3 pop3; /* POP3 コントロールブロック */
ap_init()
{
    ER ercd;
        ・
        ・
    /* POP3 の初期化 */
    ercd = pop3_ini(&pop3, 0, pop3_server, NULL);
}
```

pop3_command

[機能] POP3 コマンドの処理

[形式] ER pop3_command(T_POP3 *pop3, char *command);

pop3	POP3 コントロールブロックのポインタ
command	送信コマンド文字列（NULL）

[戻値]

E_OK	正常終了
E_PAR	不正コマンド
E_OBJ	POP3 シーケンスエラー
E_NOEXS	POP3 が未初期化
E_TMOUT	通信エラー
E_CLS	サーバーからの切断要求

[解説] POP3 コマンドを実行します。この関数の実行後に必ず pop3_rcv_reply を呼び出してください。

[コマンド]

コマンド名	パラメータ	内容
USER	メールボックス名	メールボックス名の送信
	（例）USER norti +OK password required for norti	
PASS	パスワード	メールボックスパスワードの送信
	（例）PASS abcdefg +OK norti has xxx messages	
QUIT	なし	通信終了
	（例）QUIT +OK Pop server at hostB signing off	
STAT	なし	maildrop情報の問合わせ
	（例）STAT +OK norti has xxx messages in yyy octets. xxx: メール件数	
LIST	メッセージ番号	各メールのサイズを取得
	（例）LIST 1 +OK 1 xxx xxx: メール NO.1 のサイズ	
RETR	メッセージ番号	メッセージのダウンロード要求
	（例）RETR 1 +OK xxx octets	
DELE	メッセージ番号	メールの削除要求
	（例）DELE 1 +OK Message 1 has been deleted	
NOOP	なし	サーバーが動作しているか確認
	（例）NOOP +OK	
TOP	メッセージ番号、行数	ヘッダ及び指定行数のダウンロード要求
	（例）TOP 1 1 +OK Message follows =メールの先頭の 10 行分が返る=	
UIDL	メッセージ番号	メッセージunique-idの問合せ
	（例）UIDL 1 +OK 1 xxx xxx: unique-id	

[例] 一連のシーケンス例

```
pop3_mail()
{
    ER ercd;
        ・
        ・
    /* ユーザー名の送信 */
    strcpy(command, "USER test");
    ercd = pop3_command(&pop3, command);
    ercd = pop3_rcv_reply(&pop3, reply, REPLY_BUFSZ, &sts);

    /* ユーザーパスワードの送信 */
    strcpy(command, "PASS test");
    ercd = pop3_command(&pop3, command);
    ercd = pop3_rcv_reply(&pop3, reply, REPLY_BUFSZ, &sts);

    /* メール番号１を受信する */
    strcpy(command, "RETR 1");
    ercd = pop3_command(&pop3, command);
    ercd = pop3_rcv_reply(&pop3, reply, REPLY_BUFSZ, &sts);

    /* メールデータの受信を行います */
    for(;;)
        ercd = pop3_rcv_dat(&pop3, reply, REPLY_BUFSZ);

    ercd = pop3_command(&pop3, "QUIT");
    /* QUIT コマンドのリプライはありません */
        ・
        ・
}
```

pop3_rcv_reply

[機能] コマンド実行後のリプライの受信

[形式] ER pop3_rcv_reply(T_POP3 *pop3, VP buf, int size, BOOL *sts);

pop3	POP3 コントロールブロックのポインタ
buf	リプライメッセージを入れる領域へのポインタ
size	受信バッファの領域サイズ
sts	リプライ結果(TRUE:+OK, FALSE:-ERR)

[戻値]

正の値	受信したリプライメッセージのサイズ
0	データ終結(接続が正常切断された)
E_PAR	不正コマンド
E_OBJ	POP3 シーケンスエラー
E_NOEXS	POP3 未初期化
E_TMOUT	通信エラー
E_CLS	サーバーからの切断要求

[解説] POP3 コマンド実行後サーバーより返ってくるリプライメッセージを受信します。
実際に返ってきたデータが size よりも大きい場合、のこりのデータは捨てられます。
この関数は pop3_command を実行後、実行してください。

pop3_rcv_dat

[機能] メールデータの受信

[形式] ER pop3_rcv_dat(T_POP3 *pop3, VP buf, int size);

pop3	POP3 コントロールブロックのポインタ
buf	受信データを入れる領域へのポインタ
size	受信バッファの領域サイズ

[戻値]

正の値	受信したデータのサイズ
0	データ終結
E_OBJ	POP3 シーケンスエラー
E_NOEXS	POP3 が未初期化
E_TMOUT	通信エラー
E_CLS	サーバーからの切断要求

[解説] メールデータを受信します。この関数は戻り値が 0 になるまで繰り返してください。

pop3_set_opt

[機能] POP3 オプションの変更

[形式] ER pop3_set_opt(T_POP3 *pop3, INT optname, const VP optval, INT optlen);

pop3	POP3 コントロールブロックのポインタ
optname	オプション名
optval	オプション
optlen	オプションの長さ

[戻値]

E_OK	正常終了
E_PAR	パラメータエラー

[解説] POP3 のオプションを設定します。

オプション名 optname には次の値を指定できます。

POP3_SET_IPADDR

POP3 サーバーの IP アドレスを変更します

optval　　UW 型(4byte)の IP アドレスが格納されたポインタ

optlen　　4

例) pop3_set_opt(&pop3, POP3_SET_IPADDR, &pop3_ipaddr, 4);

POP3_SET_NIF_BYNAME

POP3 が使用するネットワーク I/F をネットワーク I/F 名から設定します

optval　　ネットワーク I/F 名

optlen　　未使用(0 指定)

例) pop3_set_opt(&pop3, POP3_SET_NIF_BYNAME, "ppp", 0);

POP3_SET_NIF_BYCH

POP3 が使用するネットワーク I/F をチャンネル番号から設定します

optval　　チャンネル番号

optlen　　未使用(0 指定)

例) pop3_set_opt(&pop3, POP3_SET_NIF_BYCH, 1, 0);

※POP3_SET_NIF_BYNAME および POP3_SET_NIF_BYCH はマルチチャネル版プロトコルスタック使用時のみ指定可能です。

第５章　MIME サポート機能

5.1 特長

MIME サポート機能は送信するメールデータを作成および受信したメールデータの解析を行う機能です。添付ファイル付きのメールファイルの作成や、受信メールから添付ファイルを抽出したりすることができます。添付ファイルは Base64 を使用してエンコード／デコードされます。メールデータの保存リソースはメモリまたはファイルのいずれかを使用できます。

Version1.10 以降ではメールデータの作成と送信を同時に行う機能をサポートしました。

5.2 未サポート機能

・UUENCODE フォーマット

・Quoted-Printable フォーマット

5.3 使用する構造体

以下は MIME サポート機能で使用する構造体です。MIME サポート機能を使用する場合はメモリまたはファイルの資源を使用します。これらの資源はアプリケーションの管理化で使用されますので、各関数を実行する前に各リソースの情報をあらかじめ設定する必要があります。メモリを使用する場合はオーバーフローに注意してください。

・メールコントロールブロック

メールの基本データを格納する構造体です。メールデータのヘッダ部分、ボディ部(本文)、添付ファイルに分かれています。

```
typedef struct {
    T_MAIL_HEAD hinf;          メールヘッダ情報
    T_MAIL_FORM md1;           メール情報 1
    T_MAIL_FORM md2;           メール情報 2
    T_MAIL_FORM atta[MAX_MAIL_ATTACH];  添付ファイルフォーム
    T_MAIL_CB   cb;            内部で使用する管理ブロック
    UB ctyp;                   文字タイプ
                                   CT_USASCII      US ASCII フォーマット
                                   CT_ISO2022JP    ISO2022 日本語
    UB attnum;                 添付ファイル数
} T_MAIL;
```

ヘッダ情報や本文に日本語が使用されている場合は、CT_ISO2022JP を文字タイプに指定してください。

・メールヘッダ情報

メールヘッダ情報を格納する構造体です。メール分割の際にはこれらのデータを格納する領域をアプリケーション側で用意する必要があります。

```
typedef struct {
    B *subject;     メールのタイトルのバッファポインタ
    B *from;        発信人のバッファポインタ
    B *to;          受取人のバッファポインタ
    B *cc;          カーボンコピーのバッファポインタ
    B *date;        メールの日付へのバッファポインタ
    B *option;      オプションへのバッファポインタ
} T_MAIL_HEAD;
```

cc, to に複数のアドレスを指定する場合は','で区切ってください。

例 ：addr1@xxx.yyy, addr2@xxx.yyy

・フォーム情報

メールデータのヘッダ部分、ボディ部(本文)、添付ファイル部の詳細情報を設定する構造体です。

```
typedef struct {
  UB type;              データ形式
                            MTYP_MEM：メモリを使用
                            MTYP_FILE：ファイルを使用
                            MTYP_COM：TCP 送信
  VP        vp;         データが格納されているバッファポインタ(MTYP_MEM 設定時)、
                        またはファイルポインタ(MTYP_FILE 設定時)、
                        または SMTP コントロールブロックのポインタ(MTYP_COM 設定時)
  UH        size;       データ形式がメモリの場合はバッファサイズを指定
  char      name;       ファイル名(メール作成時に指定するファイル名)
  UB        etyp;
} T_MAIL_FORM;
```

5.4 コンフィグレーション

コンフィグレーションヘッダnonmcfg.hを#include する前に、次の様な定数を記述する
ことにより、コンフィグレーションが行えます。
() はデフォルト値で指定がない場合に使用されます。

#define SMTP_TMO	(15000/MSEC)	SMTP の通信タイムアウト
#define POP3_TMO	(15000/MSEC)	POP3 の通信タイムアウト
#define SMTP_CLIENT_PORT	(TCP_PORTANY)	SMTP クライアントポート番号
#define POP3_CLIENT_PORT	(TCP_PORTANY)	POP3 クライアントポート番号

※ TCP_PORTANY は TCP 内部で自動的に設定される設定です
※ Version.2.00 以降 このコンフィグレーションヘッダが追加されました。nonetc.h と同様にこのヘッダを必ずアプリケーションのどこかで include してください。

mime_mak_mail

[機能] メールデータ(ファイル)の作成

[形式] ER mime_mak_mail(T_MAIL *mcb);

mcb　　メールコントロールブロックへのポインタ

[戻値] E_OK　　正常終了
E_NOMEM　メモリ不足
E_OBJ　　ファイル読み出しエラー

[解説] メールデータ(またはファイル)を作成します。
添付ファイルがある場合、添付ファイルは自動的に Base64 でエンコードされます。

[例]

```
/* ファイルを使用してメールを作成する */
T_MAIL m;
func_file()
{
    T_MAIL_FORM mdata;
      ・
      ・
    /* ヘッダ情報設定 */
    m.hinf.subject = mmf_subject;
    m.hinf.from    = mmf_from;
    m.hinf.to      = mmf_to;
    m.hinf.cc      = mmf_cc;
    m.hinf.date    = NULL;
    m.hinf.option  = NULL;

    /* 作成されたメールデータの保存先 */
    m.md1.type = MTYP_FILE;
    m.md1.dp = fopen(mmf_outp_file, "w"); /* 保存するファイルを指定 */
    m.md2.type = MTYP_FILE;
    m.md2.dp = fopen(mmf_body_file, "r"); /* メール本文を指定 */

    /* 添付するファイルを設定 */
    for (i = 0; i < attnum; i++){
        m.atta[i].type = MTYP_FILE;
        m.atta[i].name = mmf_atta_file[i];
        m.atta[i].dp = fopen(m.atta[i].name, "rb"); /* 添付するファイルを指定 */
    }

    /* メールファイルの作成 */
    ercd = mime_mak_mail(&m);

    /* ファイルを閉じる */
    for (i = 0; i < m.attnum; i++)
        fclose(m.atta[i].dp);
    fclose(m.md2.dp);
    fclose(m.md1.dp);
}
```

```
/* メモリを使用してメールを作成する */
T_MAIL    m;
func_mem()
{
                .
                .
    errcd = pget_mpl(mplid, 80, &m.h.subject);
    errcd = pget_mpl(mplid, 80, &m.h.from);
    errcd = pget_mpl(mplid, 80, &m.h.to);
    errcd = pget_mpl(mplid, 80, &m.h.cc);
    errcd = pget_mpl(mplid, 80, &m.h.date);

    /* 作成されたメールデータの保存先 */
    m.md1.type = MTYP_MEM;
    m.md1.size = 10240;
    ercd = pget_mpl(mplid, m.md1.size, &m.md1.dp);

    /* ボディ部の設定 */
    m.md2.type = MTYP_MEM;
    m.md2.dp = (VP)body;
    m.md2.size = strlen(body);

    /* 添付ファイルの設定 */
    m.attnum = 1;
    m.atta[0].type = MTYP_MEM;
    m.atta[0].dp   = (VP)attach1;
    m.atta[0].size = strlen(attach1);
    m.atta[0].name = "atta1.txt";

    /* メールデータの作成 */
    ercd = mime_mak_mail(&m);
    /* m.md1.dp にデータが保存される */
    /* m.md1.size にデータサイズが保存される */
      .
      .
      .
}
```

```
/* メールを作成と送信 */
T_MAIL m;
func_com()
{
    ER ercd;
    /* ヘッダ情報設定 */

    m.hinf.subject = "Test Mail Subject";
    m.hinf.from    = "¥"Test1 mail¥" <test1@xxx.xxx>";
    m.hinf.to      = "¥"Test2 mail¥" <test2@xxx.xxx>";
    m.hinf.cc      = "¥"Test3 mail¥" <test2@xxx.xxx>";
    m.hinf.date    = NULL;
    m.hinf.option  = NULL;

    /* 作成されたメールデータのタイプ指定 */

    m.md1.type = MTYP_COM; /* サーバーへ送信する */
    m.md1.dp   = &smtp;    /* SMTP コントロールブロックのポインタ */
    m.md1.size = 0;

    /* ボディ部の設定 */

    m.md2.type = MTYP_MEM;
    m.md2.dp = (VP)body;
    m.md2.size = strlen(body);

    /* 添付ファイルの設定 */

    m.attnum = 2;
    m.atta[0].type = MTYP_MEM;
    m.atta[0].dp = (VP)sample_htm;
    m.atta[0].size = 931;
    m.atta[0].name = "sample.html";

    m.atta[1].type = MTYP_MEM;
    m.atta[1].dp   = (VP)power_gif;
    m.atta[1].size = 3236;
    m.atta[1].name = "power.gif";

    /* SMTP シーケンス開始 */

    ercd = smtp_command(&smtp, "HELO NORTi");
    ercd = smtp_command(&smtp, "MAIL <test0@xxx.xxx>");
    ercd = smtp_command(&smtp, "RCPT <test1@xxx.xxx>");

    /* メールデータの作成と送信 */

    ercd = smtp_command(&smtp, "DATA");
    ercd = mime_mak_mail(&m);
    ercd = smtp_command(&smtp, ".");

    /* メール送信完了 */

    ercd = smtp_command(&smtp, "QUIT");
}
```

mime_get_hinf

[機能] 受信したメールデータのヘッダ情報を取得

[形式] ER mime_get_hinf(T_MAIL_FORM *mdata, T_MAIL *mdb);

mdata　受信したメールデータ
mdb　メールコントロールブロックへのポインタ

[戻値] E_OK　正常終了
E_OBJ　ファイル読み出しエラー

[解説] メールデータのヘッダ情報を獲得します。取得したヘッダ情報は mdb.hinf に格納されます。

[例]

```
T_MAIL m;
func()
{
    T_MAIL_FORM mdata;
        ・
        ・
    ercd = mime_get_hinf(&mdata, &m);
    /* ヘッダから取得した Subject */
    sprintf(buf, "Subject = %s¥r¥n", m.hinf.subject);
      ・
    /* ヘッダから取得した From */
    sprintf(buf, "From    = %s¥r¥n", m.hinf.from);
        ・
    /* ヘッダから取得した To */
    sprintf(buf, "To      = %s¥r¥n", m.hinf.to);
        ・
    /* ヘッダから取得した Cc */
    sprintf(buf, "Cc      = %s¥r¥n", m.hinf.cc);
        ・
    /* ヘッダから取得した Date */
    sprintf(buf, "Date    = %s¥r¥n", m.hinf.date);
        ・
        ・
}
```

mime_get_afname

[機能] 受信したメールデータのヘッダ情報と添付ファイル情報を取得

[形式] BOOL mime_get_afname(T_MAIL_FORM *mdata, char *fname, int nlen, T_MAIL *mdb);

mdata	受信したメールデータ
fname	添付ファイル名を格納するバッファへのポインタ
nlen	添付ファイル名を格納するバッファのサイズ
mdb	メールコントロールブロックへのポインタ

[戻値] E_OK 正常終了
E_OBJ ファイル読み出しエラー

[解説] 受信したメールデータから添付ファイル名と添付ファイルの情報を取得します。
受信したメールに添付ファイルがある場合は sep_mail_data を呼び出す前に必ずこの関数を実行する必要があります。

[例]

```
T_MAIL m;
func()
{
    T_MAIL_FORM mdata;
    char fname[MAX_FIELD_LENGTH];
        .
        .
    /* メールデータ(m)から添付ファイル名 fname を取得 */
    for (n=0; mime_get_afname(&mdata, fname, sizeof fname, &m); n++){
        /* リソースはメモリを使用 */
        if (resource == MEMORY){
            /* リソース確保 */
            errcd = pget_mpl(mplid, 1024, &m.atta[n].dp);
            m.atta[n].type = MTYP_MEM;
            m.atta[n].size = 1024;
        }
        /* リソースはファイルを使用 */
        else if (resource == FILE){
            /* リソース確保 */
            m.atta[n].type = MTYP_FILE;
            m.atta[n].dp = fopen(fname[n], "wb");
        }
    }
        .
        .
        .
}
```

mime_sep_mail

[機能] メールデータ(またはファイル)を分割

[形式] ER mime_sep_mail(T_MAIL_FORM *mdata, T_MAIL *mdb);

mdata　受信したメールデータへのポインタ
mdb　メールコントロールブロックへのポインタ

[戻値] E_OK　正常終了
E_NOMEM　メモリ不足
E_OBJ　ファイル読み出しエラー

[解説] 受信したメールを各パートに分割します。添付ファイルがある場合は、この関数の実行後に必ず mime_get_afname を呼び出してください。
Base64 の添付ファイルが付加されている場合は自動的にデコードされます。

[例]

```
T_MAIL m;
func()
{
    T_MAIL_FORM mdata;
    char fname[MAX_FIELD_LENGTH];

    /* 保存するヘッダ部の設定 */
    m.md1.type = MTYP_MEM;
    errcd = pget_mpl(mplid,1024,&m.md1.dp);
    m.md1.size = 1024;

    /* 保存するボディ部の設定 */
    m.md2.type = MTYP_MEM;
    errcd = pget_mpl(mplid,1024,&m.md2.dp);
    m.md2.size = 1024;

    /* 受信したメールデータを指定 */
    mdata.type = MTYP_MEM;
    mdata.dp = data;
    mdata.size = size;

    /* 添付ファイルがある場合は添付ファイルの情報を設定 */
    if (atta_num){
        for (; mime_get_afname(&mdata, fname, sizeof fname, &m); n++){
            m.atta[n].type = MTYP_MEM;
            errcd = pget_mpl(mplid,1024,&m.atta[n].dp);
            m.atta[n].size = 1024;
        }
    }

    /* メールファイルを分割する */
    ercd = mime_sep_mail(&mdata, &m);
        ・
        ・
        ・
}
```

MailSystem for NORTi ユーザーズガイド

株式会社ミスポ http://www.mispo.co.jp
〒213-0002 神奈川県川崎市高津区二子 5-1-1 高津パークプラザ 3F
一般的なお問い合せ sales@mispo.co.jp
技術サポートご依頼 norti@mispo.co.jp