4D Internet Commands

4D Internet Commands

はじめに

4D Internet Commandsは、LANやWANで通信を行うためのユーティリティを、4Dユーザに提供します。ここ数年で、多くの人々や会社がインターネットに接続するようになりました。インターネットに接続する人々が増えるのに比例して、ビジネスコミュニティでもネットへの接続要求が増加しています。

4D Internet Commandsのコマンドを使用して、データベース開発者はインターネットのの多くのサービスにアクセスできるようになります。SMTPコマンドには、データベースから自動でメールを送信するためのコマンドが含まれます。同様に、POP3やIMAPコマンドを使用して、メールボックスからメールを取得し、データベースに格納したり、転送、自動返信、リモート検索などを実行できます。FTPコマンドでは、ファイルの転送や取得、FTPボリュームのディレクトリリストの取得などが行えます。そしてTCPとUDPコマンドは、インターネットに関連するタスクを実行するための低レベルなツールを提供します。

SMTP (Simple Mail Transfer Protocol) はインターネットで最も使用されるメール転送プロトコルです。4D Internet Commandsを使用して、ユーザはSMTPサーバ経由でメールを素早く構築し送信することができます。メールの構築と配送は一つのコマンドで可能です。メール配送のニーズがより複雑な場合、メッセージヘッダや本文、および添付ファイルをコントロールすることも可能です。インターネットのメールを使用すれば、ほぼすべての人々に情報を届けることが可能です。他にSMTPが利用できるシーンは以下のようなものがあります:

- 自動的なデータベースレポートの配信
- 自動的なメール転送データベースの作成
- メーリングリストの管理
- リモートデータベースの更新と同期

SMTPコマンドのほか、4D Internet Commandsには POP3 (Post Office Protocol, Version 3) や IMAP (Internet Message Access Protocol) メールサーバに接続し、メールメッセージや添付ファイルを取得するためのコマンドも含まれます。一連のSMTP、POP3そして IMAPコマンドはMIME標準をサポートしているので、バイナリの添付ファイルも簡単にダウンロードし保存できます。

さらにコマンドにはBinhex、Base64、AppleSingle、AppleDoubleなどさまざまな形式で添付ファイルをエンコードする機能も含まれます。

FTP (File Transfer Protocol) コマンドは、テキストやバイナリファイルをFTPサーバとやり取りするための簡単なメカニズムを提供します。FTPの一連のコマンドを使用して、リモートのボリュームをナビゲートするインタフェースを作成することができます。FTPコマンドを使用すれば、リモートボリュームをマウントするクライアントアプリケーションを使わずに、ドキュメント追跡アプリケーションを作成できます。

Transmission Control Protocol/Internet Protocol (TCP/IP) はインターネット上でデータをやり取りする際にもっとも使用されるプロトコルです。4D Internet Commandsは生のTCPパケットを送受信するためのコマンドを提供します。TCPコマンドは、独自のインターネット通信をコントロールするツールを作成することを可能にします。さらにTCP_Openコマンドでは SSL (Secured Socket Layer) プロトコルを使用した通信も可能です。

- 独自のTelnetインタフェース作成
- リモートマシンのシェルコマンド実行
- World Wide Webからドキュメントの取得
- 複数のオンラインデータベースの検索
- リモートサーバとのデータベース同期処理
- 宅配便配送情報追跡
- HTTPSを使用したWebサーバへの接続

Note: 互換性を拡張するため、4D Internet CommandsではPOP3、IMAPまたはFTP接続参照を直接低レベルのTCPコマンドに渡すことができます。またその逆も可能です。詳細は 低レベルルーチン, 概要 の節を参照してください。

UDP (User Datagram Protocol) はコネクションレスのプロトコルで、TCPよりも素早く簡単にデータの送受信ができますが、信頼性は低くなります。このプロトコルは素早い転送のニーズが高い場合に使用されます。たとえば、データのストリーミングなどです。UDPコマンドを使用して、データ送受信の基本的なツールを作成できます。

インストールおよび最低動作環境

インストール

"4D Internet Commands"プラグインは、他のプラグインと同様の方法で4Dに統合することができます。

4DアプリケーションをEasy Installオプションでインストールすると、プラグインが自動で**Plugins**フォルダにインストールされるため、4D Internet Commandsもすぐに使い始めることができます。

プラグインのインストールと設定に関する詳細は4D製品のインストールガイドを参照してください。

最低動作環境

4Dで4D Internet Commandsを使用するための最低動作環境を説明します。以下に示す条件すべてに適合する必要はなく、使用するコマンドに関連する条件を満たしていれ ば結構です。TCP/IPプロトコルやTCP/IPプロトコルスタックを使用した通信のすべての外部呼び出しは、すべての機能で必要とされます。

システム

システムの最低動作環境は4Dと同じです。詳細は4Dインストールガイドを参照してください。

4D (Mac & Windows)

- バージョン12以降のMacバージョンの4D
- バージョン12以降のWindowsバージョンの4D

(12.1より提供される) 64-bitの4D Serverには、バージョン12.1以降の4D Internet Commandsをインストールしなければなりません。

BSD, Winsock

4D Internet Commandsのすべてのコマンドは TCP/IPプロトコルを使用して通信を行います。このコマンドを実行するすべてのコンピュータは、TCP/IPド ライバがインストールされ、ユニークなIPアドレスやその他が正しく設定されていなければなりません。ほとんどのOSではTCP/IPドライバはプリイン ストールされています。たとえばMacintoshにはBSD Socketsが、WindowsにはWinsockがインストールされています。

TCP/IPの設定については、ネットワーク管理者に問い合わせてください。

ネットワークへの接続

4D Internet Commandsのコマンドを使用するには、TCP/IPをサポートするネットワークにアクセスできなければなりません。

SSL

4D Internet Commandsはバージョン12.1より、電子メールサーバーとの接続やメッセージ送信時にSSLプロトコルを使用することができます。4DICでこのプロトコルを使用するにあたり特別な設定は必要ありません。

注: 4DIC 12.1のSSL実装には陰解法 (implicit method) が使用されています。

ドメインネームサーバ

4D Internet Commandsの多くのコマンドは、ドメインネームサーバへのアクセスを必要とします。詳細はネットワーク管理者に問い合わせてください。

SMTPメールサーバ

SMTPコマンドを使用してメールを送信するには、送信者が (POP3サーバへのメール転送を行う) SMTPメールサーバにアクセスできなければなりません。

POP3メールサーバ

POP3コマンドを使用するには、POP3メールサーバのアカウントが必要です。

IMAPメールサーバ

IMAPコマンドを使用するには、IMAPメールサーバのアカウントが必要です。

用語解説

この節では、マニュアル中で使用される用語を解説します。用語は簡単に解説され、特に用語になじみがない方のために書かれています。"引数フォーマット"の章に含まれる専門用語の節では、4D Internet Commands共通の引数フォーマットについて説明しています。

NIC: "Network Information Center"。インターネットは誰か特定の団体により統制されているものではありません。中央管理団体がなければ、利用や発展が管理されているわけでも ありません。しかし、ドメイン名やIPアドレスなど、一つの管理団体によってなされるべき基本的な管理の必要性は依然として存在します。NICはこのよう な管理タスクを行うグループです。

RFC: "Request for Comments"。4D Internet Commandsの大部分はインターネット通信を処理するために定義された標準仕様に従っています。インターネットで使用される標準のメソッド、定義、お よびプロトコルはRFCとして知られるドキュメントで定義されます。**Appendix D, 追加情報** にはRFCドキュメントが参照可能ないくつかのWebサイトが紹介されています。4D Internet Commandsを使用すれば、これらのドキュメントを参照する必要はほとんどありませんが、独自のローレベルなTCP`ルーチンを開発する場合、これら に慣れる必要があります。

TCP/IPアドレス、ホスト名、およびドメイン名: IPアドレスは世界のどこかにある**特定の**マシンへの参照です。IPアドレスはピリオドで区切られた４つの数字を含む文字列です (例 "207.94.15.3")。それぞれの数字部分は 0から255までを指定できます。数学的な計算を行うことで、IPアドレスを倍長整数値に変換することができます。このドキュメントではこの倍長整数値を**ip_LongInt**として参照します。

団体がコンピュータをインターネットに接続するためには、それらのIPアドレスがネットワーク上の他のマシンと重複しないようにしなければなりません。団体は彼らのネットワークを**NIC**に登録し、**ドメイン名**を取得します。**ドメイン名**はインターネットアドレスを記憶しやすくする方法を提供します。ドメイン名はDomain Name System (DNS) により数値アドレス (Internet Protocol [IP] numbers) に変換されます。この仕組みにより、より読みやすい"www.4D.com" や "ftp.4D.com"などの利用が可能となります。

ドメイン名 = "4D.com"

ホスト名 (コンピュータ名)	=	**IPアドレス**	=	**ip_LongInt**
"www.4D.com"	=	"207.94.15.3"	=	-815919357

ホスト名とIPアドレスの対応はDNS (Domain Name System) として知られるデータベースに格納されています。DNSはお互いにドメイン名リストの変更や追加などの情報をやり取りします。TCP/IPコントロールパ ネルでは、すべてのドメイン名を解決するためにコンピュータが参照するDNSを指定します。

すべてのドメイン名は対応するIPアドレスを持つことを理解することが重要です。しかしすべてのIPアドレスが対応するドメイン名を持つわけではあ りません。また"jsmith@4D.com"のような"メールアドレス"は特定のコンピュータやIPアドレスを指し示すわけでもありません。メールアド レスは、"4D.com"を解析して得られるIPアドレスにより、配送先を指定します。メールはPOP3サーバに配送され、そのサーバは自身のユーザ "jsmith"のメールを保持します。

ドメイン名: ドメイン名はインターネット上のコンピュータを特定あるいは検索するために使用されるアドレスです。ドメイン名を使用すれば、インターネットアドレスを覚 えやすい文字列のまま使用できます。ドメイン名はDomain Name System (DNS) により、数値のインターネットアドレス (Internet Protocol [IP] numbers) に変換されます。ドメイン名は階層的であり、ドメイン名が使用するエンティティタイプに関する情報を伝達します。ドメイン名は単にドメインを表すラベルで あり、全体的なドメイン名前空間のサブセットです。階層中同じレベルのドメイン名はユニークでなければなりません。たとえば階層のトップレベルに`com`は唯一つしか存在できず、階層の次レベルに`4D.com` も唯一つしか存在できません。あなたの組織名が"CompanyName"である場合、ドメイン名"CompanyName.com"を登録し、メールア ドレス"UserName@CompanyName.com"を持つことができます。あなたの顧客はWebブラウザで"www.companyName.com"にアクセスして、組織のWebサイトにアクセスできます。

Domain Name System (DNS): ドメイン名をInternet Protocol (IP) アドレス変換するために使用される、公開されたデータベース。ドメイン名は人間にとって覚えやすく、またIPアドレスはインターネット上でコンピュータが お互いを見つけるために使用されます。ドメイン名を扱う管理者は、このデータベースの特定の部分を管理し、そのデータはインターネット上の他のユーザやコ ンピュータで利用可能となります。

エンコーディング: エンコーディングとは、同じ文字セットをサポートしないかもしれない他のコンピュータにファイルを送信するために、ファイルを別のフォーマットに変換する ことです。もっとも一般的なエンコーディングはbinary-hexadecimal (Binhex) です。Binhexエンコーディングはメッセージに追加する添付のデフォルトエンコーディングです。エンコーディングは元のファイルよりも大きなファイル を生成しますが、データフォークやリソースフォーク、Finder情報など

を文字ファイルに変換できます。4D Internet CommandsはBinhex、Base64、AppleSingle、AppleDouble、UUEncode、そしてMacBinaryなど一般的なエンコーディング方法をサポートします。

暗号化: 暗号化はメッセージの内容を読めなくするために使用されます。メッセージはPGPなどの外部プログラムを使用して暗号化し、メッセージのプライバシーを確保します。暗号化されたメッセージを読むためには、解読しなければなりません。4D Internet Commandsには暗号化の機能は用意されていません。

圧縮: 圧縮はファイルが使用する容量を減らすために使用されます。ファイルを圧縮するには、Stuffit Deluxe™?やCompact Pro™、WinZip™などのアプリケーションを使用します。圧縮されたファイルを解凍するには、専用のアプリケーションを使用します。ファイルを圧縮 アプリケーションで圧縮すると、多くの場合元のファイル名に特定の拡張子が付加されます。以下に圧縮アプリケーションと、対応する一般的な拡張子を示します。

Filename.SIT - Stuffit アプリケーション
Filename.CPT - Compact Pro アプリケーション
Filename.DD - Disk Doubler アプリケーション
Filename.ZIP - Winzip アプリケーション
Filename.SEA - 自己解凍アーカイブ、このファイルはMacintoshの独立したアプリケーションで、解凍のためのコードが含まれており、ユーザがダブルクリックする と、自分自身を解凍します。この追加のコードのため、自己解凍アーカイブはFilename.SITやFilename.CPTで圧縮された場合に比べ、 ファイルサイズが大きくなります。しかし解凍時、ユーザは専用の解凍アプリケーションを必要としないため、エンドユーザにとって利点があります。

圧縮した後でも転送の際には、マシンからマシンに正しくファイルが転送されることを確実にするために、エンコーディングが必要であることに注目してください。

引数フォーマット

ここでは、このマニュアルを通して使用される主要なパラメーターの意味やフォーマットを説明します。

パラメーター	型		説明
hostName	文字列	→	ホスト名 (例: "www.companyname.com") IP アドレス (例: "204.118.90.2")
ip_LongInt	倍長整数	→	IPアドレスの倍長整数参照
mailAddress	テキスト	→	例: "jsmith@4d.com"
addressList	テキスト	→	例: "jsmith@4d.com, jdupont@4d.fr" または "jsmith@4d.com"+Char(13)+"jdupont@4d.fr"
localPath	テキスト	→	- ドキュメント Mac: "My Hard Drive:4DDB:SalesDB:Report" Win: "C:¥MyDrive¥4DDB¥SalesDB¥Report.txt" - ディレクトリ Mac: "My Hard Drive:CoolStuff:" Win: "C:¥MyDrive¥CoolStuff¥"
hostPath	テキスト	→	- ドキュメント "/usr/jsmith/reports/salesreport.txt" - ディレクトリ "/usr/jsmith/reports/"
tcp_ID	倍長整数	→	開かれたTCPセッション参照
smtp_ID	倍長整数	→	新規メールメッセージ参照
pop3_ID	倍長整数	→	開かれたPOP3セッション参照
imap_ID	倍長整数	→	開かれたIMAP接続参照
ftp_ID	倍長整数	→	開かれたFTPセッション参照
udp_ID	倍長整数	→	UDPソケット参照
戻り値	整数	←	エラーコード

hostName

hostName は"dns.4d.com" や "204.118.90.2"のようなホスト名またはIPアドレスです。ホスト名はドメイン名システムを使用して解決されます。プライマリおよびセカンダリDNSは通常OSのコントロールパネルを使用して設定されます。*hostName* を引数として受け取る4DICのコマンドはホスト名、あるいはIPアドレス形式いずれのフォーマットも受け入れますが、サーバーのIPアドレスは変更されることがあるため、ホスト名形式を使用することが推奨されます。

ip_LongInt

ホスト名はドメイン名システムを使用してIPアドレスに変換されます。IPアドレスは文字であらわされますが、計算を行うことで倍長整数に変換することができます。**NET_NameToAddr** や **NET_AddrToName** などのコマンドを使用すればこの変換処理を実行させることができます。この倍長整数値をこのマニュアルでは *ip_LongInt* という名前で参照します。倍長整数値は直接的にTCP接続を行う特別な状況下でのみ利用されます。またホスト名の代わりに解決済みの倍長整数値を使用することで、ディスクスペースを節約できます。しかしIPV6との互換性のため、倍長整数値は使用すべきではありません。

mailAddress

MailAddress は "user_name@domain_name" という形式のメールアドレスです。このドキュメントでは *mailAddress* はひとつのメールアドレスを表します。メールアドレスのリストは *addressList* で参照します。引数が *mailAddress* のみであ

る場合、ひとつのメールアドレスしか引数として渡せないことを意味します。*mailAddress* のフォーマットはユーザー名とドメイン名両方を含んでいなければなりません:

- "Felix Unger" <felix@pristine.com>
- oscar@slobs.com (Oscar Madison)

addressList

addressList は一つ以上のメールアドレスを含みます。各メールアドレスはカンマまたはキャリッジリターンで区切ります。以下の例題はいずれも有効なアドレスリストを生成します:

```
$AddressList:="jsmith@4d.com"
$AddressList:="jsmith@4d.com,scott@4d.com,marcel@4d.fr"
For($i;1;Size of array(aAddresses))
   $AddressList:=$AddressList+aAddresses{$i}+"\r"
End for
```

localPath

localPath はユーザーマシン上におけるファイルやディレクトリの場所です。MacOSではコロンをディレクトリ区切り文字として使用します。たとえば"My Hard Drive"ハードディスク内の"Reports"フォルダーにある"My Report.txt"は"My Hard Drive:Reports:My Report.txt"というパス名であらわすことができます。MacOSにおいて、ディレクトリのパスはコロンで終了しなければなりません。たとえば新しいレポートを同じフォルダーに保存するには"My Hard Drive:Reports:"と指定します。ファイルとディレクトリいずれを参照するかは、コマンドが呼び出されるコンテキストに基づきます。Windowsでも同様のルールを用いますが、ディレクトリ区切り文字にはバックスラッシュ (¥　日本語フォント環境では円マークになることがあります) を使用します。

hostPath

hostPath はUnix OSで実行されているコンピューター上のファイルやディレクトリを示す場所です。Unix環境ではディレクトリ区切り文字としてスラッシュ (/) を使用します。たとえば"4D"ディレクトリ中の"reports"ディレクトリの中の"report.txt"ファイルは"/4D/reports/report.txt"で示すことができます。ディレクトリパス名は "/" で終了しなければなりません。パス名先頭の"/"はボリュームのルートを表します。

smtp_ID, pop3_ID, imap_ID, ftp_ID, tcp_ID

4D Internet Commandsは多くのコマンドで"ID"を引数にとります。各通信機能のコマンドセットは独自のセッションを確立し、それは"ID"で参照できます。セッションを開くコマンドに続く一連のコマンドはこのIDを使用してセッションを特定します。

各通信コマンド (SMTP, POP3, IMAP, FTP, TCP, UDP) で取得できるIDを他の通信コマンドに渡すことはできません。しかしPOP3, IMAP, FTP接続のIDは低レベルのTCPコマンドに渡すことができます。詳細は**低レベルルーチン - 概要**を参照してください。

セッション参照	セッションを開く	セッションを閉じる
tcp_ID	TCP_Open または TCP_Listen	TCP_Close
smtp_ID	SMTP_New	SMTP_Clear
pop3_ID	POP3_Login	POP3_Logout または POP3_VerifyID
imap_ID	IMAP_Login	IMAP_Logout または IMAP_VerifyID
ftp_ID	FTP_Login	FTP_Logout または FTP_VerifyID
udp_ID	UDP_New	UDP_Delete

戻り値

すべての4D Internet Commands (**IT_ErrorText** と **IT_Version**を除く) は関数の結果として整数値を返します。この整数値には4Dデータベースに通知すべき関数の結果としてエラーコードが含まれます。コマンドの実行に成功すると、0が返されます。非0が返された場合、エラーが発生したことを意味します。4D Internet Commandsのエラーコードについては**Appendix C, 4D Internet Commandsエラーコード**を参照してください。

IC IMAP メール閲覧

- IMAP4 コマンド - 概要
- IMAP_Capability
- IMAP_CloseCurrentMB
- IMAP_CopyToMB
- IMAP_CreateMB
- IMAP_Delete
- IMAP_DeleteMB
- IMAP_Download
- IMAP_GetCurrentMB
- IMAP_GetFlags
- IMAP_GetMBStatus
- IMAP_GetMessage
- IMAP_GetPrefs
- IMAP_ListMBs
- IMAP_Login
- IMAP_Logout
- IMAP_MsgFetch
- IMAP_MsgInfo
- IMAP_MsgLst
- IMAP_MsgLstInfo
- IMAP_MsgNumToUID
- IMAP_RenameMB
- IMAP_Search
- IMAP_SetCurrentMB
- IMAP_SetFlags
- IMAP_SetPrefs
- IMAP_SubscribeMB
- IMAP_UIDToMsgNum
- IMAP_VerifyID

IMAP4 コマンド - 概要

IMAPコマンドを使用してIMAPメールサーバと通信し、メッセージを処理することができます。IMAPコマンドはRFC2060で示されたInternet Message Access Protocol, Version 4 revision 1 (`IMAP4rev1`) と互換があります。`IMAP4 rev1`は、メールボックスと呼ばれるリモートのメッセージフォルダを、ローカルのメールボックスと同様に管理できるようにします。

IMAPコマンドを使用してメールボックスの作成、削除名称変更などの操作、新着メッセージのチェック、メッセージの削除、メッセージフラグの設定やクリア、メッセージの検索、選択したメッセージの受信などができます。

用語

"接続"はネットワークの接続開始 (*IMAP_Login*) から接続終了 (*IMAP_Logout*) までの、IMAPクライアント/サーバの一連のシーケンスを指します。

"セッション"はメールボックスが選択 (*IMAP_SetCurrentMB*) されてから、選択終了 (*IMAP_SetCurrentMB*, *IMAP_CloseCurrentMB*) または接続の終了 (*IMAP_Logout*) までの、クライアント/サーバの一連のシーケンスを指します。

IMAP接続概要

- TCP接続の初期化 (必要に応じて): *IT_MacTCPInit* (PPP接続の場合、*IT_PPPConnect*コマンドを*IT_MacTCPInit*よりも前に呼び出さなければなりません)。
- 接続を開く: *IMAP_Login*
- メールボックスの管理: リスト, 作成, 削除, 名称変更, 購読/購読解除, ステータスの取得など。
- メールボックスを指定してセッションを開く: *IMAP_SetCurrentMB*。カレントのメールボックスを設定した後、メッセージを管理できます。
- メッセージの管理: リスト, メッセージのダウンロードや削除、メッセージフラグのリストや更新、他のメールボックスへのコピー、(ダウンロードをしない) メッセージの検索や受信など。
- カレントのメールボックスメッセージの操作が終了したら、セッションを閉じるか、他のメールボックスをカレントのメールボックスにできます。いずれの場合も、IMAPサーバは恒久的にメッセージを更新します。例えば、削除フラグが設定されたメッセージを削除します。
- すべての操作が終了したらログアウトします。接続を閉じる: *IMAP_Logout*。
- 他の操作: 環境設定、接続のチェック、IMAPサーバ上の自動切断タイマーのリセット

IMAP コマンドテーマ

IMAP関連のコマンドは二つのセクション**IC IMAP メール閲覧**と**IC ダウンロードしたメール**に分けられます。この分離はメールを読み込む異なる方法を表しています。IMAPサーバからメールを読みだす際、メッセージ (またはメッセージに関する情報) は (変数, フィールド, 配列など) 4Dのオブジェクトに格納されるか、ディスクにダウンロードされます。このセクションでは、4D Internet Commandsを使用してIMAPサーバからメッセージを読み込む能力について説明します。

メッセージの受信に二つの方法が用意されている理由は、大きな情報をダウンロードするかもしれない場合のメモリの制限のためです。例えば5MBの添付ファイルがあるメッセージは容易にデータベースの格納容量をオーバーフローさせます。4Dオブジェクトでこのサイズを格納できるのはピクチャや Blobです。しかしメッセージや添付ファイルをこれらのオブジェクトに変換するのは大きなメモリ空間が必要なため、効率的ではありません。これを解決するために、このセクションには*IMAP_Download*コマンドが用意され、IMAPサーバからメッセージをローカルディスクにダウンロードできるようになっています。

ディスクに保存された後は、**IC ダウンロードしたメール**セクションのコマンドを使用して、ファイルを操作できます。

メールボックスのメカニズム

IMAPのメールボックスはフォルダのように操作でき、フォルダがファイルやサブフォルダを含むことができるように、メー

ルボックスもメッセージやサブメールボックスを含むことができます。

メールボックスには完全な階層名を使用してアクセスできます。それぞれの階層レベルは、*IMAP_ListMBs* コマンドで返される階層セパレータで分けられます。

セパレータを使用して、子メールボックスを作成したり、名前階層の上位レベルまたは下位レベルを検索したりできます。階層のトップレベルのすべての子階層は、同じセパレータ文字を使用します。

Note: メッセージはカレントのメールボックスを選択 (*IMAP_SetCurrentMB*) した後にのみ操作できます。

アカウントごとに一つまたは複数のメールボックスを持つことができます。

メールボックスは大文字小文字を区別しますが、大文字と小文字の違いしかない複数のメールボックスを作成することはできません。

INBOXメールボックスは特別なケースです。このメールボックスはすべてのアカウントに存在し、受信したメッセージを格納するために使用されます。アカウントがセットアップされると、INBOXは自動で作成されます。

ユーザはINBOXメールボックスを削除できませんが、名称変更は可能です。新しい空のINBOXがすぐに作成されます。
INBOXは大文字小文字を区別しません。

メッセージ総数や新規メッセージ数などのいくつかのメールボックス属性は、メールボックスがカレントでなくてもチェックできます。

msgNumとuniqueID

IMAPコマンド使用時、よく利用される引数、特にメールボックスメカニズムの*msgNum*と*uniqueID*について理解することが重要です。

*msgNum*は、*IMAP_SetCurrentMB*コマンドが実行された時のメールボックス中のメッセージ番号です。

カレントメールボックスを選択すると、メールボックス中のメッセージには1からメッセージ数までの番号がふられます。メッセージには受信された順番 に番号がふられ、1がもっとも古いメッセージとなります。メッセージふられた番号は、カレントのメールボックスを選択してから (*IMAP_SetCurrentMB*) それを閉じるまで (*IMAP_CloseCurrentMB*, *IMAP_SetCurrentMB*または *IMAP_Logout*) のみ有効です。

メールボックスが閉じられると、削除のマークが付けられたメッセージは削除されます。

ユーザがIMAPサーバに再ログインすると、メールボックス中のメッセージは再度1から番号付けされます。例えばメールボックスに10のメッセージ があり、1から5までが削除されると、再度メールボックスを開いた際には以前の6から10のメッセージに1から5の番号が付けられます。

例えば以下の例を見てみましょう: IMAPサーバにログインして以下のメッセージリストを取得したとします:

msgNum	uniqueID	日付	送信者	件名
1	10005	1 Jul 2001 ...	danw@acme.com	Sales lead...
2	10008	1 Jul 2001 ...	frank@acme.com	Site-License order
3	10012	3 Jul 2001 ...	joe@acme.com	Lunch anyone?
4	20000	4 Jul 2002 ...	kelly@acme.com	Your wife called...
5	20001	4 Jul 2002 ...	track@fedex.com	FedEx tracking

このセッションの間、メッセージ番号3と4が削除されます。カレントのメールボックスを閉じると、実際に削除が実行されます。サーバに再度ログインすると、メッセージリストは以下のようになります:

msgNum	uniqueID	日付	送信者	件名
1	10005	1 Jul 2001 ...	danw@acme.com	Sales lead...
2	10008	1 Jul 2001 ...	frank@acme.com	Site-License order
3	20001	4 Jul 2002 ...	track@fedex.com	FedEx tracking

msgNum はスタティックな値ではなく、セッションごとに異なる可能性があります。メッセージ番号は、メールボックスが選択された時のメッセージの状態で変化します。

ところで、*uniqueID*はユニークな値で、IMAPサーバが昇順にメッセージに付加します。メールボックスにメッセージが追加

されると、以前に追加されたメッセージより上の値が割り当てられます。

残念なことに、IMAPサーバはメッセージの主たる参照として*uniqueID*を使用しません。IMAPコマンドを使用する際は、メッセージを参照するために*msgNum*を使用する必要があります。データベースにメッセージ参照を格納し、メッセージそのものはサーバに残すようなソリューションを構築する際、開発者はこの点に留意する必要があります。

推奨

IMAPは全体として相対的なものであり、独立してテストすることはできないため、"すべてをテストする"ことをお勧めします。アカウントを取得可能なすべてのサーバに対してテストを行うようにしてください。

詳細は以下のサイトをチェックしてください:

- IMAP Products and Services: http://www.imap.org/products.html
- MailConnect: http://www.imc.org/imc-mailconnect.

POP3とIMAP4コマンドの比較

Login	同等	IMAPにaPOP引数はありません
VerifyID	同等	
Delete	同等	IMAPはリアルタイムで削除を行います。POP3はPOP3_Logoutを待って恒久削除を行います。IMAP_SetFlagsで¥Deletedフラグを使用すれば、POP3_Deleteコマンドと同じ結果になります。
Logout	同等	
SetPrefs	同等	IMAPにattachFolderはありません。POP3のattachFolderはオプションです。
GetPrefs	同等	SetPrefsのattachFolderに関するメモを参照
MsgLstInfo	同等	
MsgInfo	同等	
MsgLst	同等	
UIDToMsgNum	同等	IMAPのmsgUIDは倍長整数ですが、POP3のmsgUIDは文字列です。
Download	同等	
POP3_Reset	代替	IMAP_Searchで¥Deletedの検索を行い、IMAP_SetFlagsで¥Deletedフラグを削除します。
POP3_BoxInfo	代替	IMAP_SetCurrentMBとIMAP_MsgLstInfoを使用する必要があります。
IMAP_ MsgNumToUID	代替	
GetMessage	殆ど同等	IMAPはよりパワフルで、"ボディのみ"を選択可能
POP3_Charset	同等無	IMAPは自動で文字セットを管理
IMAP_Capability	同等無	IMAPプロトコルのみ
IMAP_ListMBs	同等無	IMAPプロトコルのみ
IMAP_GetMBStatus	同等無	IMAPプロトコルのみ
IMAP_SetCurrentMB	同等無	IMAPプロトコルのみ
IMAP_GetCurrentMB	同等無	IMAPプロトコルのみ
IMAP_CloseCurrentMB	同等	IMAPプロトコルのみ

	無		
IMAP_CopyToMB	同等無	IMAPプロトコルのみ	
IMAP_SubscribeMB	同等無	IMAPプロトコルのみ	
IMAP_CreateMB	同等無	IMAPプロトコルのみ	
IMAP_DeleteMB	同等無	IMAPプロトコルのみ	
IMAP_RenameMB	同等無	IMAPプロトコルのみ	
IMAP_SetFlags	同等無	IMAPプロトコルのみ	
IMAP_GetFlags	同等無	IMAPプロトコルのみ	
IMAP_Search	同等無	IMAPプロトコルのみ	
IMAP_MsgFetch	同等無	IMAPプロトコルのみ	

Notes:

- IMAPとPOP3サーバ: IMAPサーバの場合、*msgID*は倍長整数であることに注意してください。
- POP3とIMAPプロトコルでは、削除の方法が異なります。*IMAP_Delete*はリアルタイムでメッセージを取り除きます。*POP3_Delete*と同じ結果を得るには、*IMAP_SetFlags*を使用して**¥Deleted**フラグを設定してください。*POP3_Reset*と同じ結果を得るには、*IMAP_SetFlags*を使用して**¥Deleted**を削除します。

- 4D Internet commandsではPOP3, IMAP, FTP接続参照を直接低レベルTCPコマンドに渡すことができ、またその逆も可能です。詳細は**低レベルルーチン - 概要**セクションを参照してください。

IMAP_Capability

IMAP_Capability (imap_ID ; capability) -> 戻り値

引数	型		説明
imap_ID	倍長整数	➡	IMAPログイン参照
capability	テキスト	⬅	IMAPの機能
戻り値	整数	⮌	エラーコード

説明

*IMAP_Capability*コマンドは、IMAPサーバがサポートする機能名をスペース区切りで返します。このリストにはIMAPのバージョンや、拡張、`IMAP4rev1` プロトコルへの修正など、IMAPサーバがサポートする機能が返されます。

4D Internet CommandsがIMAP4に対応していることを確認するためには、`IMAP4rev1`が*capability*テキストに含まれていなければなりません。

IMAP_CloseCurrentMB

IMAP_CloseCurrentMB (imap_ID) -> 戻り値

引数	型		説明
imap_ID	倍長整数	➡	IMAP参照
戻り値	整数	⮌	エラーコード

説明

IMAP_CloseCurrentMB コマンドは、他のメールボックスを選択せずまた*IMAP_Logout*を実行せずに、カレントの処理対象のメールボックスを閉じます。**IMAP_CloseCurrentMB** は**¥Deleted** フラグがセットされたメッセージを恒久的に取り除きます。

Note: IMAPでは、クライアント/サーバモードで同時に同じメールボックスに対し処理を行うことを許可しています。誰かが同期を実行し、接続が開かれたままの場合、最後に使用されたメールボックスは選択されたままとなります。サーバの実装によっては、たとえユーザが"非接続モード"で作業していたとしても、このメールボックスを他のセッションで使用するユーザは有効な情報を取得できず、正しく操作することができません。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

IMAP_CopyToMB

IMAP_CopyToMB (imap_ID ; startMsg ; endMsg ; mbNameTarget ; msgDelete) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
startMsg	倍長整数	⇒	開始メッセージ番号
endMsg	倍長整数	⇒	終了メッセージ番号
mbNameTarget	テキスト	⇒	コピー先メールボックス名
msgDelete	整数	⇒	0= 元のメールボックスからメッセージを削除しない, 1= 元のメールボックスからメッセージを削除する
戻り値	整数	⇐	エラーコード

説明

IMAP_CopyToMB コマンドは*startMsg*から*endMsg*までのメッセージを、*mbNameTarget* で指定したメールボックスの最後にコピーします。メッセージのフラグと内部日付は通常保持されます (IMAPサーバの実装による)。

コピーされた後、メッセージは元のメールボックスから削除されません。コピー元のメッセージを削除したい場合、以下の方法のいずれかを使用できます:

- *IMAP_Delete*コマンドを使用する
- *msgDelete* オプション引数に1を指定する
- *IMAP_SetFlags* (**¥Deleted**)を使用する: メッセージはセッションが閉じられたときに削除されます。

Note: *msgDelete* 引数は*IMAP_Delete*を実行します; ゆえに削除対象には*startMsg*と*endMsg*の間のメッセージおよび**¥Deleted**フラグが設定されたメッセージが含まれます。

コピー先メールボックスが存在しない場合、エラーが返されます。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

startMsg はコピーするメッセージの開始位置を示すメッセージ番号です。メッセージ番号は*imap_ID*で参照されるメールボックスのすべてのメッセージリスト中のメッセージの位置を示す番号です。

endMsg はコピーするメッセージの終了位置を示すメッセージ番号です。メッセージ番号は*imap_ID*で参照されるメールボックスのすべてのメッセージリスト中のメッセージの位置を示す番号です。

Note: *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* そして*IMAP_CopyToMB*コマンドは、*startMsg*が*endMsg*よりも大きい場合でもエラーを返しません。この場合、コマンドは何も行いません。

mbNameTarget はメッセージのコピー先を指定するメールボックス名です。

msgDelete オプション引数を使用して、コピー元メールボックスからメッセージを削除するか指定できます。

- 0= 削除しない (デフォルト)
- 1= 削除する

*msgDelete*が省略された場合、デフォルト値が使用されます。

コピーに失敗すると、コピー元のメールボックスからメッセージは削除されません。

メッセージを削除する権限がユーザにない場合、エラーメッセージが生成されます。

IMAP_CreateMB

IMAP_CreateMB (imap_ID ; mbName) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
mbName	テキスト	⇒	作成するメールボックス名
戻り値	整数	⇐	エラーコード

説明

IMAP_CreateMB コマンドは指定された名称でメールボックスを作成します。IMAPサーバの階層区切り文字がメールボックス名に含まれる場合、IMAPサーバは必要に応じて親メールボックスを作成します。

例えば、階層区切り文字が"/"のとき、“Projects/IMAP/Doc”を作成しようとすると、

- "Projects"と"IMAP"が既に存在すれば"Doc"メールボックスのみが作成されます。
- "Projects"が既に存在すれば"IMAP"と"Doc"が作成されます。
- いずれも存在しなければ"Projects"、"IMAP"、"Doc"メールボックスが作成されます

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

mbName は作成するメールボックス名です (IMAPの命名規則を参照してください)。

Note: INBOX (この名称はサーバユーザの主たるメールボックス名として予約されています) または既存のメールボックスを作成しようとすると、エラーが生成されます。

IMAP_Delete

IMAP_Delete (imap_ID ; startMsg ; endMsg) -> 戻り値

引数	型		説明
imap_ID	倍長整数	➡	IMAPログイン参照
startMsg	倍長整数	➡	開始メッセージ番号
endMsg	倍長整数	➡	終了メッセージ番号
戻り値	整数	⮌	エラーコード

説明

IMAP_Delete コマンドは、*startMsg*から*endMsg*の範囲のメッセージに**¥Deleted**フラグを設定し、**¥Deleted**フラグが設定されたメッセージを削除します (現在のセッションですでに**¥Deleted**フラグが設定されていたメッセージを含む)。削除はIMAPサーバで実行され、接続を閉じる (*IMAP_Logout*) または他のメールボックスを選択する (*IMAP_SetCurrentMB*) またはカレントのメールボックスを閉じる (*IMAP_CloseCurrentMB*) 時にも行われます。

すぐに削除したくない場合、*IMAP_SetFlags* コマンドを使用して、後ほど削除するメッセージに**¥Deleted**フラグを設定します。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

startMsg は削除するメッセージの開始メッセージ番号です。

endMsg は削除するメッセージの終了メッセージ番号です。

Note: *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* そして*IMAP_CopyToMB*コマンドは、*startMsg*が*endMsg*よりも大きい場合でもエラーを返しません。この場合、コマンドは何も行いません。

IMAP_DeleteMB

IMAP_DeleteMB (imap_ID ; mbName) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
mbName	テキスト	⇒	削除するメールボックス名
戻り値	整数	⇐	エラーコード

説明

IMAP_DeleteMB コマンドは指定されたメールボックスを恒久的に削除します。INBOXや存在しないメールボックスを削除しようとするとエラーが生成されます。

IMAP_DeleteMB コマンドは、子メールボックスを持つメールボックスや、**¥Noselect**メールボックス属性が設定されたメールボックスを削除できません。

子メールボックス名を指定した場合でかつ**¥Noselect**メールボックス属性が指定されていない場合、メールボックスを削除できます。この場合、メールボックス中のメッセージはすべて削除され、**¥Noselect**メールボックス属性が設定されます。

Note: いくつかのサーバでそれが可能であるにしても、IMAPプロトコルは空でないメールボックスの削除を保証していません。またカレントのメールボックスが開かれている間は、そのメールボックスを削除すべきではありません。まずそのメールボックスを閉じるべきです。いくつかのサーバはカレントのメールボックスの削除を許可しません。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

mbName は削除するメールボックス名です。

IMAP_Download

IMAP_Download (imap_ID ; msgNum ; headerOnly ; fileName ; updateSeen) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
msgNum	倍長整数	⇒	メッセージ番号
headerOnly	整数	⇒	0 = メッセージ全体, 1 = ヘッダのみ
fileName	テキスト	⇒	ローカルファイル名
		⇐	実際のローカルファイル名
updateSeen	整数	⇒	0 = ¥Seen フラグを追加; 1= ¥Seen フラグを追加しない
戻り値	整数	⇐	エラーコード

説明

IMAP_Download コマンドは、IMAPサーバからメッセージをディスク上のファイルとして取得するために使用します。添付ファイルが含まれるメッセージやサイズが32Kを超えるIMAPメッセージは、このコマンドを使用してダウンロードすべきです。添付ファイルはこの方法でダウンロードされたメッセージからのみ取り出せます。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

msgNum はメールボックス中の取り出すメッセージを指定するメッセージ番号です。*msgNum* は現在処理対象のメールボックス中のすべてのメッセージにおける、メッセージの位置を表します。特定のメッセージのメッセージ番号がセッションをまたいで同一であるとは限りません。

headerOnly は、メッセージ全体をダウンロードするかヘッダのみをダウンロードするかを指定する整数値です。

fileName には、メッセージを保存する際に使用するファイル名と、そのパスを渡します。この値は以下の三つの方法で指定できます:

- "": *IMAP_SetPrefs*で設定したフォルダに、“temp1” (同じ名前のファイルが存在する場合、未使用のファイル名が見つかるまで“temp2”, “temp3”などが試されます) などのファイル名でファイルを保存します。
- "FileName": *IMAP_SetPrefs*で設定したフォルダに、FileNameで保存されます。
- "Path:FileName" = FileNameで指定したパスにファイルを保存します。

最初の二つの場合、*IMAP_SetPrefs*でフォルダが指定されていなければ、ストラクチャと同階層 (4D シングルユーザの場合) または4Dクライアントフォルダ (4D Serverの場合) に保存されます。

ファイルがディスクに保存されると、*fileName* 引数に渡された変数に最終的なファイル名が返されます。既に存在するファイル名を*fileName*に指定して*IMAP_Download*をコールすると、増分された数値がファイル名に付加され、*fileName*変数に返されます。

updateSeen は**¥Seen**フラグをメッセージに付加するかしないかを指定する整数値です:

- 0 = **¥Seen** フラグを追加する
- 1 = **¥Seen** フラグを追加しない

この引数はオプションで、指定しない場合のデフォルト値は0です。

IMAP_GetCurrentMB

IMAP_GetCurrentMB (imap_ID ; mbName) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
mbName	テキスト	⇐	カレントメールボックス名
戻り値	整数	⇒	エラーコード

説明

IMAP_GetCurrentMB コマンドはカレントのメールボックス名を返します。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

mbName にはカレントメールボックスの完全名が返されます。*mbName*の値が空の文字列の場合、メールボックスは選択されていません。

IMAP_GetFlags

IMAP_GetFlags (imap_ID ; startMsg ; endMsg ; msgFlagsArray ; msgNumArray) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
startMsg	テキスト	⇒	開始メッセージ番号
endMsg	テキスト	⇒	終了メッセージ番号
msgFlagsArray	文字配列	⇐	メッセージ毎のフラグ値
msgNumArray	倍長整数配列	⇐	メッセージ番号の配列
戻り値	整数	⇐	エラーコード

説明

IMAP_GetFlags コマンドは指定したメッセージのフラグリストを返します。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

startMsg は情報を取得するメッセージリストの開始メッセージ番号です。メッセージ番号は現在処理対象のメールボックス中のすべてのメッセージにおける、メッセージの位置を表します。

endMsg は情報を取得するメッセージリストの終了メッセージ番号です。メッセージ番号は現在処理対象のメールボックス中のすべてのメッセージにおける、メッセージの位置を表します。

Note: *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* そして*IMAP_CopyToMB*コマンドは、*startMsg*が*endMsg*よりも大きい場合でもエラーを返しません。この場合、コマンドは何も行いません。

msgFlagsArray はテキスト配列で、*startMsg*から*endMsg*の間のメッセージのフラグリスト (空白区切り) が返されます。

msgNumArray 倍長整数配列で、*startMsg*から*endMsg*の間のメッセージ番号が返されます。

IMAP_GetMBStatus

IMAP_GetMBStatus (imap_ID ; mbName ; msgNber ; newMsgNber ; unseenMsgNber ; mbUID) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
mbName	テキスト	⇒	メールボックス名
msgNber	倍長整数	⇐	指定したメールボックス中のメール数
newMsgNber	倍長整数	⇐	¥Recentフラグがセットされたメッセージ数
unseenMsgNber	倍長整数	⇐	¥Seenフラグが設定されていないメッセージ数
mbUID	倍長整数	⇐	指定したメールボックスのユニークなID
戻り値	整数	⇒	エラーコード

説明

IMAP_GetMBStatus コマンドは、*mbName*で指定したメールボックスのステータス値を返します。このコマンドはカレントのメールボックスを変更しません (*IMAP_SetCurrentMB*参照) 、また指定されたメールボックスのステータスにも影響しません (特にメッセージの **¥Recent** フラグを失うようなことはありませんが、これはIMAP4サーバの実装にもよります)。

このコマンドは特に以下のケースで有効です:

- メールボックスのユニークIDをチェックまたは取得する
- メールボックスのセッションを開かずに、新着や未読のメッセージ数をチェックする

重要: カレントのメールボックスに対して*IMAP_GetMBStatus* コマンドをコールしないようにすることを強く推奨します。これを行うと問題が発生し、返された情報が不必要にカレントのメールボックスのステータスと同期されてしまうかもしれません (特に新着メール)。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

mbName はステータスを取得する既存のメールボックスの完全名です。

Note: *IMAP_ListMBs* コマンドと異なり、*mbName* 引数はワイルドカードを受け入れません。

msgNber にはカレントメールボックスのメッセージ数が返されます (コマンドが呼び出されるとき0に設定され、エラーの際には-1が返されます)。

newMsgNber にはカレントメールボックスの新規メッセージ数が返されます (コマンドが呼び出されるとき0に設定され、エラーの際には-1が返されます)。

unseenMsgNber にはカレントメールボックスの未読メッセージ数が返されます (コマンドが呼び出されるとき0に設定され、エラーの際には-1が返されます)。

mbUID にはメールボックスのユニークIDが返されます (コマンドが呼び出されるとき0に設定され、エラーの際には-1が返されます)。

IMAP4プロトコルでは、メールボックス名はメールボックスを指定する目的で使用するには十分でありません。そこで、ユニークIDがメールボックスに割り当てられています。このIDは特に同期処理に有効です。

メールボックス"A"が"B"に名称変更されていたり、削除されていた場合、ユニークIDを使用してチェックができます。

また、このIDを使用してメールボックス"A"が削除された後に、新しくメールボックス"A"が作成されたかもチェックできます。

IMAP_GetMessage

IMAP_GetMessage (imap_ID ; msgNum ; offset ; length ; msgPart ; msgText ; updateSeen) -> 戻り値

引数	型		説明
imap_ID	倍長整数	→	IMAPログイン参照
msgNum	倍長整数	→	メッセージ番号
offset	倍長整数	→	受信を開始する文字のオフセット値
length	倍長整数	→	受信する文字数
msgPart	整数	→	0 = メッセージ全体, 1 = ヘッダのみ, 2= ボディのみ
msgText	テキスト	←	メッセージテキスト
updateSeen	整数	→	0 = ¥Seenフラグを更新する; 1 = 更新しない
戻り値	整数	↩	エラーコード

説明

IMAP_GetMessage コマンドは、*IMAP_SetCurrentMB*で参照されるカレントのメールボックス中、*msgNum*で指定したメッセージの完全なテキストを返します。*IMAP_SetPrefs*で指定しない限り、メッセージ中のラインフィードは取り除かれます。

IMAP_GetMessage コマンドは*msgPart*引数の指定に基づき、メッセージ全体、ヘッダのみ、あるいはボディのみを返します。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

msgNum は受信するメッセージを指定するメッセージ番号です。この番号はカレントのメッセージリスト中の位置を表します。セッションをまたいだ場合、この番号は特定のメッセージに対して変わることがあります。

*offset*は、*msgPart*で指定したパートで、読み込みを開始する文字位置を表す数値です。

*length*は*offset*位置からの読み込む文字数を指定する数値です。この数値は32,000以下でなければなりません。*msgPart*のサイズが32Kを超えるメッセージは*IMAP_Download* コマンドを使用してディスクに保存なければなりません。

*msgPart*は受信するメッセージのパートを指定します:

- 0 = メッセージ全体
- 1 = ヘッダのみ
- 2 = ボディのみ (最初のtext/plainが見つかったところ)

メッセージ全体またはヘッダのみを取得した場合、デコードは行いません。他方、ボディのみを取得した場合、必要に応じてテキストはデコードされ変換されます (デコードと変換のルールについては*POP3_Charset*の説明を参照)。

updateSeen はメッセージに**¥Seen**フラグを追加するかどうかを指定します。この引数はオプションで、指定しない場合デフォルト値が使用されます。

- 0 = **¥Seen**フラグを追加 (デフォルト)
- 1= **¥Seen**フラグを追加しない

*msgText*には受信したテキストが返されます。

IMAP_GetPrefs

IMAP_GetPrefs (stripLineFeed ; msgFolder) -> 戻り値

引数	型		説明
stripLineFeed	整数	⇐	0 = ラインフィードを保持する, 1 = ラインフィードを取り除く, -1 = 変更しない
msgFolder	テキスト	⇐	メッセージフォルダパス
戻り値	整数	⇒	エラーコード

説明

IMAP_GetPrefs コマンドはIMAPコマンドの現在の環境設定値を返します。

環境設定値は引数に渡された変数に返されます。

stripLineFeed にはラインフィードの処理に関する設定が返されます。

msgFolder には受信したメッセージのデフォルトの格納場所が返されます。

IMAP_ListMBs

IMAP_ListMBs (imap_ID ; mbRefName ; mbName ; mbNamesArray ; mbAttribsArray ; mbHierarArray ; subscribedMB) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
mbRefName	テキスト	⇒	ヌル文字またはメールボックス名またはメールボックス階層レベル
mbName	テキスト	⇒	ヌル文字またはメールボックス名またはワイルドカード
mbNamesArray	文字配列	⇐	メールボックス名配列 (パス名)
mbAttribsArray	文字配列	⇐	メールボックス属性配列
mbHierarArray	文字配列	⇐	階層区切り文字配列
subscribedMB	整数	⇒	0 = 利用可能なすべてのユーザメールボックスをリスト、1 = 購読済みメールボックスのみをリスト
戻り値	整数	⇒	エラーコード

説明

IMAP_ListMBs コマンドは、接続ユーザの利用可能なメールボックスのリストと付随する情報を返します。コマンドの実行に失敗すると、配列は初期化されます。

返されるメールボックスリストは、*mbRefName*と*mbName*二つの引数の組み合わせによって決まるため、これら二つの引数をひとつとして考える必要があります。

*subscribedMB*が1の場合、返されるリストは購読済みのメールボックスに制限されます (*IMAP_SubscribeMB*を参照) 。

多くのメールボックスや複雑な階層のメールボックスストラクチャのスキャンを行うなどのため、*IMAP_ListMBs*の実行に時間がかかる場合、以下の方法をとることができます:

- *IMAP_ListMBs*にワイルドカードを使用する (後述)
- *IMAP_ListMBs* コマンド使用時に*subscribedMB* 引数を1にして、*IMAP_SubscribeMB* コマンドを使用して設定したメールボックスのみをリストする

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

mbRefName は *mbName* 引数とともに、検索対象のメールボックスを指定します。参照名 (*mbRefName*) はUnixシステムのCurrent Working Directoryとして使用されるべきです。言い換えれば、メールボックス名 (*mbName*) は参照名 (*mbRefName*) によって特定されるディレクトリ内に置かれたファイルとして解釈されます。IMAPの仕様は、参照名 (*mbRefName*) の解釈を"処理系に依存"としていることに注意してください。*mbRefName* 参照引数を使用しない動作モードをユーザに提供することを強く推奨します。そうすれば、参照引数を使用しない古いサーバと相互運用することが可能となります。

mbRefName が空の文字列なら、*mbName* 引数のみがメールボックスリストのために使用されます。

mbRefName がメールボックス名またはメールボックス階層レベルを含んでいる場合、この引数は*mbName* 引数がどのように解釈されるかを決定するために使用されます。

Note: 参照引数を使用する場合、階層区切り文字を置くことを強く推奨します。これにより、どのようなIMAPサーバが使用されていても、完全に準拠することが保障されます。

mbName は *mbRefName* 引数とともに使用され、*mbName* 引数が解釈されるコンテキストを決定します。

mbName が空の文字列の場合、階層区切り文字が返されます。

Note: *mbRefName* 引数を使用してbreakout facilityを実装する場合、メールボックス引数に前置階層区切り文字を使用するかしないかを、ユーザが選択できるようにすべきです。なぜならサーバにより、さらには同じサーバのメールストアにより前置階層区切り文字の処理が異なるからです。ある時は前置階層区切り文字は“参照引数の切り捨て”を意味し、またある時は二つは結合され、追加の階層区切り文字を切り捨てることを意味するからです。

mbNamesArray 配列は利用可能なメールボックスの名前を受け取ります。

mbAttribsArray 配列は利用可能なメールボックスの属性を受け取ります。

メールボックス属性

四つのメールボックス属性が定義されています:

- **¥Noinferiors**: 子レベル階層が現在存在せず、作成することもできない。
- **¥Noselect**: この名前を選択可能なメールボックスとして使用できない。
- **¥Marked**: サーバがメールボックスに"interesting"のマーク付けをしている; おそらくメールボックスには、最後に選択されてから追加されたメッセージが含まれている。
- **¥Unmarked**: メールボックスには、最後に選択されてからメッセージが追加されていない。

mbHierarArray 配列には、利用可能なメールボックスの階層区切り文字が返されます。

階層区切り文字は、メールボックス名中の階層レベルを区切るために使用される文字です。この文字を使用して子メールボックスを作成したり、上位レベルや下位レベルを検索したりできます。トップレベルのすべての子ノードは同じ区切り文字を使用します。

subscribedMB は購読済みのメールボックスのみをリストするかどうか指定するために使用します。0を渡すと利用可能なすべてのユーザメールボックスがリストされます。1を渡すと購読済みのメールボックスがリストされます。*subscribedMB* はオプションの引数で、指定しない場合のデフォルト値は0です。

例題 1

以下の例では:

```
IMAP_ListMBs(imap_ID;"4DIC/Work/";"Test";mbNamesArray;mbAttribsArray;mbHierarArray)
```

"4DIC/Work/Test"から利用可能なすべてのメールボックスが返されます。

IMAPサーバが意図されたとおり解釈を行わない場合、*mbRefName*を使用せず、*mbRefName*と*mbName* の値を結合して、*mbName*に渡してください:

```
IMAP_ListMBs(imap_ID;"";"4DIC/Work/Test";mbNamesArray;mbAttribsArray;mbHierarArray)
```

例題 2

以下の例は:

```
IMAP_ListMBs(imap_ID;"";"";mbNamesArray;mbAttribsArray;mbHierarArray)
```

階層区切り文字を返します。

ワイルドカード文字を使用する

*mbRefName*と*mbName*引数にワイルドカード文字を使用して、メールボックスの選択をより簡単にできます。カレントのワイルドカードの例を以下に示しますが、ワイルドカードの解釈はIMAPサーバにより異なることに留意してください。つまり以下の例は動作しないことがあります。その場合、お使いのIMAPサーバのワイルドカードをチェックしてください。

- " * " はその位置の0個以上の文字にマッチします:

```
IMAP_ListMBs(imap_ID;"";"*";mbNamesArray;mbAttribsArray;mbHierarArray)
```

は接続ユーザが利用可能なすべてのメールボックスを返します。

```
IMAP_ListMBs(imap_ID;"";"Work*";mbNamesArray;mbAttribsArray;mbHierarArray)
```

はルート“Work”にマッチするすべての利用可能なメールボックスを返します。

- “ % ”は“ * ”と同じものですが、階層区切り文字にマッチしない点が異なります。“%”が*mbName* 引数の最後の文字である場合、マッチした階層レベルも返されます。これらの階層レベルが選択不可のメールボックスである場合、**¥Noselect** メールボックス属性が返されます (“メールボックス属性”の節を参照)。

```
IMAP_ListMBs(imap_ID"";"Work/%";mbNamesArray;mbAttribsArray;mbHierarArray)
```

は接続ユーザが利用可能な、ルート“Work”にマッチするすべてのメールボックスと、さらに一つの階層レベルが返します。

“%” はメールボックスの階層をレベルごとに解析する際に有効です。
以下のメールボックス階層がある時に:

```
INBOX
   MailboxA
      MailboxAA
      MailboxAB
   MailboxB
      MailboxBA
      MailboxBB
   MailboxC
      MailboxCA
      MailboxCB
```

```
IMAP_ListMBs(imap_ID;"";"%";mbNamesArray;mbAttribsArray;mbHierarArray)
```

は INBOX, MailboxA, MailboxB そして MailboxCを返します。

```
IMAP_ListMBs(imap_ID;"";"MailboxA%";mbNamesArray;mbAttribsArray;mbHierarArray)
```

は MailboxAA と MailboxABを返します。

このテクニックを使用して、

```
IMAP_ListMBs(imap_ID;"";"*";mbNamesArray;mbAttribsArray;mbHierarArray)
```

の回答待ちでユーザコントロールが長時間失われてしまうようなことを避けることができます。

IMAPサーバ自身が検索レベルを制限するかもしれないことに注意してください。

IMAP_Login

IMAP_Login (hostName ; userName ; password ; imap_ID {; sessionParam}) -> 戻り値

引数	型		説明
hostName	文字	⇒	IMAPサーバのホスト名またはIPアドレス
userName	文字	⇒	ユーザ名
password	文字	⇒	パスワード
imap_ID	倍長整数	⇐	このIMAPログインの参照
sessionParam	倍長整数	⇒	1 = SSLを使用, 0または省略 = SSLを使用しない
戻り値	整数	⇒	エラーコード

説明

IMAP_Login コマンドは、ユーザ名とパスワードを使用して、IMAPサーバにログインします。

このログインの結果、*imap_ID*に他のIMAPコマンドで使用するログイン参照が返されます。

接続は*IMAP_Logout* コマンドの実行またはIMAPサーバの無動作タイマがタイムアウトしたときに閉じられます。

hostName はIMAPサーバのホスト名またはIPアドレスです。ホスト名の利用を推奨しますが、必要であればIPアドレスを使用することもできます。

userName はIMAPサーバにログインするためのユーザ名です。

password はIMAPサーバにログインするためのパスワードです。

imap_ID は倍長整数変数で、ログイン参照が返されます。この引数には値を受け取るために、倍長整数変数を渡さなくてはなりません。この変数は、このセッションに関連するアクションを行うIMAPコマンドで使用されます。

*IMAP_Login*に失敗すると、*imap_ID*には0が設定されます。
オプションの*sessionParam*引数を使用すると、接続にSSLプロトコルを使用することができます:

- 1を渡すと、IMAPサーバーへの接続はSSLで行われます (同期モード)、
- 0を渡すかこの引数を省略すると、接続は標準の非保護モードで行われます。

例題

典型的な接続処理は以下の通りです:

```
$ErrorNum:=IMAP_Login(vHost;vUserName;vUserPassword;vImap_ID;1)
If($ErrorNum =0)
   C_TEXT(vCapability)
   $ErrorNum:=IMAP_Capability(vImap_ID;vCapability))
  ` vImap_ID引数を使用するIMAPコマンドを実行
End if
$ErrorNum:=IMAP_Logout(vImap_ID)
```

IMAP_Logout

IMAP_Logout (imap_ID) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
		⇐	0 = ログオフに成功した
戻り値	整数	⇒	エラーコード

説明

IMAP_Logout コマンドは、*imap_ID* で参照される開かれたIMAP接続からログオフします。コマンドの実行に成功すると、*imap_ID*変数に0が返されます。

Note: 接続を閉じると、自動でカレントのセッションも閉じられます。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

IMAP_MsgFetch

IMAP_MsgFetch (imap_ID ; msgNum ; msgDataItem ; msgDataItemValue) -> 戻り値

引数	型		説明
imap_ID	倍長整数	➡	IMAPログイン参照
msgNum	倍長整数	➡	メッセージ番号
msgDataItem	テキスト	➡	取得するデータ項目
msgDataItemValue	テキスト	⬅	データ項目値
戻り値	整数	↩	エラーコード

説明

IMAP_MsgFetch コマンドは、メッセージをダウンロードせずに、指定したメッセージの一つ以上の基本的なデータ項目を取得することを可能にします。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

msgNum はチェックするメッセージを指定するメッセージ番号です。この番号はカレントのメッセージリスト中の位置を表します。セッションをまたいだ場合、この番号は特定のメッセージに対して変わることがあります。

msgDataItem は取得したい一つ以上のデータ項目を指定するためのテキストです。複数のデータ項目を指定する場合、スペースを使用して各項目を区切ります。データ項目には二つの種類があります。

- 基本データ項目: 一部分の情報のみを取得する。
- マクロデータ項目: 基本データ項目から成るいくつかの部分項目の情報を一度に取得する。一般に使用されるデータ項目のセットが三つ定義されています。マクロはそれ自身でのみ使用され、他のマクロやデータ項目と一緒に使用することはできません。

データ項目については、後述の“基本データ項目”と“マクロデータ項目”の節を参照してください。

msgDataItemValue はテキスト変数で、*msgDataItem* 引数値に応じて一つの*DataItem/DataItemValue*ペア、または*DataItem/DataItemValue*ペアのリストが返されます。

- ひとつの*DataItem/DataItemValue*の場合、返されるテキストのデータ構造は次のとおりです: *DataItem name+Space+DataItemValue*
- *DataItem/DataItemValue*ペアのリストの場合、返されるテキストのデータ構造は次のとおりです: *DataItem name1+Space+DataItemValue1+Space+DataItem name2+Space+DataItemValue2*

msgDataItemValue には*msgDataItem* 引数に応じて、括弧に挟まれたリスト、クォートでくくられた文字列またはひとつの文字列が返されます。

- 括弧で挟まれたリストは以下の構造になっています (**FLAGS**の例題参照):
  (*FirstDataItemValue+space+2ndDataItemValue*)
  括弧で挟まれたリストが括弧のみを返す場合、これは項目値がないことを示します。このルールは括弧に挟まれたアドレスリストには適用されません (**ENVELOPE**参照)。
- クォートでくくられた文字列の構造は以下のとおりです (**INTERNALDATE**の例題参照): *DataItem name+Space+Quote+DataItemValue+Quote*
  DataItem が "" の場合、空の文字列を意味します。
- クォートで挟まれていない文字列は整数、倍長整数または数値を示し、構造は以下のとおりです: *DataItem name+Space+DataItemValue*
  この場合、適切なタイプに変換する必要があります (UIDの例題参照)。

Note: クォートは、文字列にスペースや括弧など特別な文字が含まれる場合に使用されます。それゆえ、**IMAP_Fetch** コマンドの結果文字列を解析する際は、文字列の内容を処理する際にクォート文字を考慮します。

基本データ項目

- **INTERNALDATE**
  IMAPサーバ上のメッセージの内部的な日付と時刻を取得します。これは"Date"ヘッダの日付と時刻とは異なります。これはメッセージを受信した日付と時刻を示します。SMTPサーバから送信されたメッセージでは、この日付は通常メッセージの最終的な配送日付と時間を反映します。**IMAP_Copy** コマンドの後に送信されたメッセージでは、このデータは通常ソースメッセージの内部的な日付と時刻を反映します。
  INTERNALDATEデータ項目値はクォートで囲まれて返されます。

例題:

```
msgDataItem:="INTERNALDATE"
$Err:=IMAP_MsgFetch(imap_ID;1;msgDataItem;msgDataItemValue)
```

msgDataItem には INTERNALDATE "17-Jul-2001 15:45:37 +0200" が返されます。

- **FLAGS**
  指定されたメッセージに設定されたフラグが、括弧に挟まれて返されます。フラグはスペースで区切られます。

例題:

```
msgDataItem:="FLAGS"
$Err:=IMAP_MsgFetch(imap_ID;1;msgDataItem;msgDataItemValue)
```

指定したメッセージにフラグが付けられていない場合、*msgDataItem* には FLAGS () が返されます。
¥Seen と **¥Answered** フラグが指定されたメッセージに設定されている場合、*msgDataItem* にはFLAGS (¥Seen ¥Answered) が返されます。

- **RFC822.SIZE**
  メッセージのバイトがRFC-822フォーマットで返されます。データ項目はスペースで区切られます。クォートでくくられた文字列が返されます。この文字列を倍長整数値に変換する必要があります (**UID**の例題を参照)。

例題:

```
msgDataItem:="RFC822.SIZE"
$Err:=IMAP_MsgFetch(imap_ID;1;msgDataItem;msgDataItemValue)
```

msgDataItem には RFC822.SIZE 99599 が返されます。

- **ENVELOPE**
  指定したメッセージのヘッダ部分が括弧つきリストで返されます。サーバは、メッセージヘッダを解析してこの値を返します。
  ヘッダフィールドは以下の順で返されます: date, subject, from, sender, reply-to, to, cc, bcc, in-reply-to, message-id。date, subject, in-reply-to そして message-id フィールドはクォート付きで返されます: ENVELOPE ("date" "subject" (from) (sender) (reply-to) (to) (cc) (bcc) "in-reply-to" "message-id")

例題:

```
msgDataItem:="ENVELOPE"
$Err:=IMAP_MsgFetch(imap_ID;1;msgDataItem;msgDataItemValue)
```

msgDataItem には ENVELOPE ("Tue, 17 Jul 2001 17:26:34 +0200" "Test" (("RSmith" NIL "RSmith" "test")) (("RSmith" NIL "RSmith" "test")) (("RSmith" NIL "RSmith" "test")) (("RSmith" NIL "RSmith" "test")) () () "" "<ee6b33a.-1@Mail.x6foadRIbnm>") が返されます。

Date:	"Tue, 17 Jul 2001 17:26:34 +0200"	date ヘッダ
Subject:	"Test"	subject ヘッダ
From:	(("RSmith" NIL "RSmith" "test"))	アドレスストラクチャ
Sender:	(("RSmith" NIL "RSmith" "test"))	アドレスストラクチャ
reply-to:	(("RSmith" NIL "RSmith" "test"))	アドレスストラクチャ
to:	(("RSmith" NIL "RSmith" "test"))	アドレスストラクチャ
cc:	()	Cc ヘッダは使用されていない
bcc:	()	Bcc ヘッダは使用されていない
in-reply-to:	""	In-reply-to ヘッダ
message-id:	"<ee6b33a.-1@Mail.x6foadRIbnm>"	message-id ヘッダ

from, sender, reply-to, to, cc そして bcc フィールドは括弧つきのアドレスストラクチャです。アドレスストラクチャは括弧つきのリストで、メールアドレスを示します。アドレスストラクチャフィールドは以下の順番です: 個人名, [SMTP] at-domain-list (ソースルート), メールボックス名、ホスト名。例えば、 (("RSmith" NIL "RSmith" "test")).

([RFC-822]) グループシンタックスはアドレスストラクチャの特別な形式で示され、ホスト名フィールドがNILになります。メールボックスフィールド名もNILの場合、これはグループ終了マーカ (RFC 822 シンタックスではセミコロン) です。メールボックス名フィールドがNILでない場合、これはグループ開始マーカで、メールボックス名フィールドはグループ名フレーズを保持します。
適用できないエンベロップのフィールドやアドレスストラクチャはNILとして表示されます。サーバはデフォルトで“from”フィールドから、reply-toとsenderフィールドを使用しなければならないことに注意してください。

- **BODY**
  拡張データ (**BODYSTRUCTURE**を参照) を除き、**BODYSTRUCTURE**と同じ情報を返します。

例題:

```
msgDataItem:="BODY"
$Err:=IMAP_MsgFetch(imap_ID;1;msgDataItem;msgDataItemValue)
```

msgDataItem には BODY ("TEXT" "PLAIN" ("CHARSET" "us-ascii") NIL NIL "8BIT" 8 1) が返されます。

- **BODYSTRUCTURE**
  メッセージのMIMEボディストラクチャを返します。サーバはメッセージヘッダのMIMEヘッダフィールドとボディパートのMIMEヘッダを解析して、結果を返します。このデータ項目は特に、メッセージをダウンロードせずにメッセージの内容をスキャンする場合に有効です。例えば、それぞれのパートのサイズを素早くチェックしたり、添付ファイルの名前を知る場合に有効です。**BODYSTRUCTURE** は括弧つきのリストで、括弧つきのリスト、クォートされた文字列、およびクォートなしの文字列を返します。
  メッセージの内容により、**BODYSTRUCTURE** は“non-multipart”括弧つきリストまたはネストしたリスト (“multipart” 括弧つきリスト) になります:
  - “**non-multipart**”括弧つきリスト: これは例えば、マルチパートでないメールと同様です。48行 2279バイトの単純なテキストメッセージは以下のようなボディストラクチャになります: ("TEXT" "PLAIN" ("CHARSET" "us-ascii") NIL NIL "8BIT" 8 1 NIL NIL NIL)。
    “non-multipart”括弧つきフィールドの基本フィールドは以下の順番になります:

ボディタイプ	メディアタイプ名を示す文字列 (Content-type: メディアタイプ 例 TEXT)
ボディサブタイプ	内容のサブタイプ名 (Content-type: サブタイプ 例 PLAIN)
ボディ引数	属性/値ペアの括弧つきリスト
括弧つきリスト	[例 ("CHARSET" "US-ASCII" "NAME" "cc.diff") "CHARSET"の値が"US-ASCII"で、"NAME"の値が"cc.diff"
ボディid	コンテンツIDを与える (他のボディへの参照を可能にする)。ボディに"Content-ID"ヘッダフィールドでラベルが付けらる。multipart/alternative メディアタイプの場合、Content-ID値には特別な意味を持たせられる。これはRFC 2046のmultipart/alternativeの節で説明されている。
ボディ説明	内容の説明を与える文字列
ボディエンコーディング	コンテンツ転送エンコーディング (Content-Transfer-Encoding)
ボディサイズ	ボディのサイズ (バイト)。これはtransfer encoding時のサイズであり、デコード後のサイズではない。

- MESSAGE タイプとRFC822サブタイプは、基本フィールドに続いて、エンベロップストラクチャ、ボディストラクチャ、カプセル化されたメッセージのテキスト行のサイズを含みます。
- TEXT タイプは、基本フィールドに続いて、テキスト行のボディサイズ、を含みます。これはtransfer encoding時のサイズであり、デコード後のサイズではないことに注意してください。拡張データは基本フィールド、そして上にあげたタイプ特有のフィールドが続きます。拡張データは**BODY**では返されません。**BODYSTRUCTURE** を使用します。

“non multipart”括弧つきリストの拡張データは、もしあれば、指定された順番でなければなりません:

ボディMD5	[MD5]で定義された、ボディのMD5文字列
ボディ配置	Disposition type文字列から成る括弧つきリストと、[DISPOSITION]で定義されたdispositionの属性名/属性値ペアの括弧つきリスト
ボディ言語	[LANGUAGE-TAGS]で定義された、ボディ言語を示す文字列または括弧つきリスト

続く拡張データはこのバージョンのプロトコルでは定義されてなく、マルチパートの拡張データ下で定義されます。
例題: ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 2279 48 NIL NIL NIL)
説明: ("bodytype" "bodysubtype" (BodyParameterParenthesizedList) bodyId bodyDescription "bodyEncoding" BodySize BodySizeInTextLines ExtensionDataBODYmd5 ExtensionDataBodyDisposition ExtensionDataBodyLanguage)

- “**multipart**”括弧つきリスト: これはマルチパートメールの場合です。これには“non-multipart”括弧つきリストが含まれます。
  ネストされた括弧は、複数のパートを示します。括弧つきリストの最初の要素はボディタイプの代わりにネストしたボディです。括弧つきリストの二番目の要素はマルチパートのサブタイプ (mixed, digest, parallel, alternative, etc.) です。
  マルチパートのサブタイプの次には拡張データが続きます。拡張データは、もしあれば、定められた順番となります。

ボディ引数	属性名/属性値の括弧つきリストの括弧つきリスト
ボディ配置	Disposition type文字列から成る括弧つきリストと、[DISPOSITION]で定義されたdispositionの属性名/属性値ペアの括弧つきリスト
ボディ言語	[LANGUAGE-TAGS]で定義された、ボディ言語を示す文字列または括弧つきリスト

続く拡張データはこのバージョンのプロトコルでは定義されていません。この拡張データは0個以上のNIL、文字列、数値、またはこれらの括弧つきリストからなります。**BODYSTRUCTURE** のフェッチを行うクライアントの実装は、そのような拡張データを受け入れるように準備されていなければなりません。サーバの実装は、このプロトコルのリビジョンにより定義されるまで、そのような拡張データを送信してはなりません。
例題: *BODYSTRUCTURE (("TEXT" "PLAIN" ("CHARSET" "us-ascii") NIL NIL "7BIT" 22 1 NIL NIL NIL)("APPLICATION" "BYTE-STREAM" ("NAME" "casta37.jpg" "X-MAC-TYPE" "4A504547" "X-MAC-CREATOR" "6F676C65") NIL NIL "BASE64" 98642 NIL ("ATTACHMENT" ("FILENAME" "casta37.jpg")) NIL) "MIXED" ("BOUNDARY" "4D_====================1385356==") NIL NIL)*
説明: *(("bodytype" "bodysubtype" (BodyParameterParenthesizedList) bodyId bodyDescription "bodyEncoding" BodySize BodySizeInTextLines ExtensionDataBODYmd5 ExtensionDataBodyDisposition ExtensionDataBodyLanguage) ("bodytype" "bodysubtype" (BodyParameterParenthesizedList) bodyId bodyDescription "bodyEncoding" BodySize BodySizeInTextLines ExtensionDataBODYmd5 ExtensionDataBodyDisposition ExtensionDataBodyLanguage) "multipartSubtype" (ExtensionDataBodyParameterList) ExtensionDataBodyDisposition ExtensionDataBodyLanguage))*

- **UID**
  メッセージのユニークなIDを取得します。これは*IMAP_UIDToMsgNum*を実行するのと同じことです。この数値はテキストエリアに返されるので、倍長整数値に変換しなければなりません。
  例題:

```
msgDataItem:="UID"
$Err:=IMAP_MsgFetch(imap_ID;1;msgDataItem;msgDataItemValue)
```

msgDataItemValue にはUID 250000186が返されます。

倍長整数に変換するには:

```
C_LONGINT(vLongint)
VLongint:=Num("250000186")
```

マクロデータアイテム

- **FAST**
  以下のマクロと同等: (FLAGS INTERNALDATE RFC822.SIZE)
  例題:

```
$Err:=IMAP_MsgFetch(imap_ID;msgNum;"FAST";msgDataItemValue)
```

msgDataItemValue には "FLAGS (¥Seen ¥Answered) INTERNALDATE "17-Jul-2001 15:45:37 +0200" RFC822.SIZE 99599" が返されます。

- **ALL**
  以下のマクロと同等: (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE)

- **FULL**
  以下のマクロと同等: (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE BODY)

IMAP_MsgInfo

IMAP_MsgInfo (imap_ID ; msgNum ; msgSize ; uniqueID) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
msgNum	倍長整数	⇒	メッセージ番号
msgSize	倍長整数	⇐	メッセージサイズ
uniqueID	倍長整数	⇐	サーバ上のメッセージのユニークID
戻り値	整数	↩	エラーコード

説明

IMAP_MsgInfo コマンドは、カレントのメールボックスにある、*msgNum*で指定したメッセージの情報を返します。メッセージのサイズおよびユニークIDが返されます。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

*msgNum*はメールボックス中の、情報を取得するメッセージのメッセージ番号を渡します。*msgNum*は現在のメッセージリスト中のメッセージの位置を表します。セッションをまたぐと、メッセージ番号は変わる可能性があります。

msgSize には、*msgNum*で参照されるメッセージのサイズが返されます。

uniqueID には、サーバ上のメッセージのユニークIDが返されます。*uniqueID*はIMAP4サーバがメッセージ割り当てる値です。これはメッセージ番号と異なり、セッションをまたいでも変わることのない値です。*uniqueID*は、サーバからメッセージを既にダウンロード済みか検証するために使用できます。

IMAP_MsgLst

IMAP_MsgLst (imap_ID ; startMsg ; endMsg ; msgHeaderArray ; msgNumArray ; msgIdArray ; msgValueArray) -> 戻り値

引数	型		説明
imap_ID	倍長整数	➡	IMAPログイン参照
startMsg	倍長整数	➡	開始メッセージ番号
endMsg	倍長整数	➡	終了メッセージ番号
msgHeaderArray	文字配列	➡	取り出すヘッダ配列
msgNumArray	倍長整数配列	⬅	メッセージ番号配列
msgIdArray	倍長整数配列	⬅	ユニークメッセージID配列
msgValueArray	2D文字列配列, 2Dテキスト配列	⬅	ヘッダ値の二次元配列
戻り値	整数	↩	エラーコード

説明

IMAP_MsgLst コマンドは、メールボックスの内容の特定の情報を取得するために使用します。このコマンドを使用して、メッセージリストの特定の列をリクエストできます。このコマンドはヘッダの項目値のみを返すことができます。ボディを取得することはできません。必要に応じてヘッダの内容はデコードされ変換されます。 (デコードと変換のルールについては*POP3_Charset*の説明を参照)。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

startMsg は情報を取得するメッセージリストの開始メッセージ番号です。メッセージ番号は現在処理対象のメールボックス中のすべてのメッセージにおける、メッセージの位置を表します。

endMsg は情報を取得するメッセージリストの終了メッセージ番号です。メッセージ番号は現在処理対象のメールボックス中のすべてのメッセージにおける、メッセージの位置を表します。

Note: *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* そして*IMAP_CopyToMB*コマンドは、*startMsg*が*endMsg*よりも大きい場合でもエラーを返しません。この場合、コマンドは何も行いません。

msgHeaderArray は取り出すメールヘッダを指定する文字配列またはテキスト配列です。

msgNumArray は、*startMsg*から*endMsg*までのそれぞれのメッセージ番号を受け取る倍長整数配列です。

msgIdArray は、*startMsg*から*endMsg*までのそれぞれのメッセージユニークIDを受け取る倍長整数配列です。

msgValueArray 二次元配列で、*msgHeaderArray*で指定されたヘッダの値を受け取ります。リクエストされたそれぞれのヘッダは、*msgValueArray*の一次元目に対応する配列をもちます。

例題

```
aHeaders{1}:="Date:"
aHeaders{2}:="From:"
aHeaders{3}:="Subject:"
IMAP_MsgLst(IMAP_ID;vStart;vEnd;aHeaders;aMsgNum;aMsgId;aValues)
```

aValues{1}{1} は “Thu, 19 November 1998 00:24:02 -0800”
aValues{2}{1} は “Jack@4d.com”
aValues{3}{1} は “Call your wife”
のようになります。

エラーは以下のように処理されます:

1. 通信に関連するエラーのみが返されます。コマンドがネットワークやシンタックス、サーバなどのエラーのため正しく実行できなかった場合、対応するエラーコードが返されます。
2. 指定された範囲のメッセージが存在しなかった場合やエラーを受け取った場合、
    - そのメッセージの配列要素は作成されません。
    - エラーコードは返されません。
3. 指定したヘッダが一部あるいはすべて、メッセージ中で全く見つからなかった場合、エラーは生成されません。
    - メッセージの配列要素は作成されます。
    - メッセージ番号配列とユニークID配列の要素には正しい値が返されます。
    - メッセージに存在しないヘッダに対しては、配列要素に空の文字列が返されます。
    - エラーコードは返されません。

IMAP_MsgLstInfo

IMAP_MsgLstInfo (imap_ID ; startMsg ; endMsg ; msgSizeArray ; msgNumArray ; msgIdArray) -> 戻り値

引数	型		説明
imap_ID	倍長整数	→	IMAPログイン参照
startMsg	倍長整数	→	開始メッセージ番号
endMsg	倍長整数	→	終了メッセージ番号
msgSizeArray	倍長整数配列	←	サイズ配列
msgNumArray	倍長整数配列	←	メッセージ番号配列
msgIdArray	倍長整数配列	←	ユニークメッセージID配列
戻り値	整数	↩	エラーコード

説明

IMAP_MsgLstInfo コマンドは (*IMAP_SetCurrentMB* コマンドで指定された) 現在処理対象のメールボックスの、一連のメッセージに関する情報を返します。情報は三つの配列に返され、配列のそれぞれの要素は一つのメッセージに対応します。メッセージサイズや番号に関する情報が返されます。配列は事前に定義されていなければなりません。*IMAP_MsgLstInfo* コマンドはそれぞれの配列サイズをメッセージ数にリセットします。

IMAP_MsgLstInfo コマンドは、カレントのメッセージリスト中のメッセージの取得に失敗しても、エラーは返しません。エラーが発生すると、そのメッセージの配列要素は作成されません。コマンドがメッセージを読み込むと、*msgNumArray*には順番に数値が格納されます。問題があった場合には、*msgNumArray*配列に格納された数値に抜けが生じます。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

startMsg は情報を取得するメッセージリストの開始メッセージ番号です。メッセージ番号は現在処理対象のメールボックス中のすべてのメッセージにおける、メッセージの位置を表します。

endMsg は情報を取得するメッセージリストの終了メッセージ番号です。メッセージ番号は現在処理対象のメールボックス中のすべてのメッセージにおける、メッセージの位置を表します。

Note: *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* そして*IMAP_CopyToMB*コマンドは、*startMsg*が*endMsg*よりも大きい場合でもエラーを返しません。この場合、コマンドは何も行いません。

sizeArray は、*startMsg*から*endMsg*までのそれぞれのメッセージサイズを受け取る倍長整数配列です。

msgNumArray は、*startMsg*から*endMsg*までのそれぞれのメッセージ番号を受け取る倍長整数配列です。

msgIdArray は、*startMsg*から*endMsg*までのそれぞれのメッセージユニークIDを受け取る倍長整数配列です。

IMAP_MsgNumToUID

IMAP_MsgNumToUID (imap_ID ; msgNum ; unique_ID) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
msgNum	倍長整数	⇒	メッセージ番号
unique_ID	倍長整数	⇐	メッセージユニークID値
戻り値	整数	⇒	エラーコード

説明

IMAP_MsgNumToUID コマンドは、*imap_ID*で参照されるカレントのメールボックス中のメッセージリストの現在のメッセージ番号を、現在の*unique_ID*に変換します。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

*msgNum*は*unique_ID*を取得する、現在のメッセージリスト中の位置を表す番号です。

unique_ID にはIMAPサーバ上で検索されたメッセージのユニークIDが返されます。

IMAP_RenameMB

IMAP_RenameMB (imap_ID ; mbName ; newMBName) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
mbName	テキスト	⇒	名称を変更するメールボックス名
newMBName	テキスト	⇒	新しいメールボックス名
戻り値	整数	⇐	エラーコード

説明

IMAP_RenameMB コマンドはメールボックスの名称を変更します。存在しないメールボックスの名称を変更しようとしたり、既存のメールボックス名と同じ名称に変更しようとしたりすると、エラーが生成されます。

Note: INBOXの名称変更は許可されていますが、この場合特別な動作をします。これを行うと、指定された名称の新しいメールボックスが作成され、INBOX内のすべてのメッセージがそのメールボックスに移動されます。INBOXは空になります。サーバの実装がINBOXの子メールボックスを許可している場合、名称変更によるこれらのメールボックスへの影響はありません。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

mbName は名称を変更するメールボックス名です (IMAPの命名規則を参照してください).

newMBName は *mbName*メールボックスに適用する新しいメールボックスの完全名です。

IMAP_Search

IMAP_Search (imap_ID ; searchCriteria ; msgNumArray) -> 戻り値

引数	型		説明
imap_ID	倍長整数	➡	IMAPログイン参照
searchCriteria	テキスト	➡	検索条件
msgNumArray	倍長整数配列	⬅	メッセージ番号配列
戻り値	整数	↩	エラーコード

説明

IMAP_Search コマンドは、カレントのメールボックスで、検索条件に合致するメッセージを検索します。*searchCriteria*は一つ以上の検索キーからなり、*msgNumArray*には検索結果のメッセージシーケンス番号が返されます。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

*searchCriteria*は一つ以上の検索キーを指定するテキスト引数です (後述の使用可能な検索キーを参照)。検索キーは一つの項目または一つ以上の検索キーの括弧つきリストです。例えば:

SearchKey1 = FLAGGED
SearchKey2 = NOT FLAGGED
SearchKey3 = FLAGGED DRAFT

Note: 通常検索は大文字小文字を区別しません。

- *searchCriteria*が空の文字列の場合、検索は“select all”と同様です:

```
IMAP_Search(imap_ID;"";msgNumArray)
```

  はカレントの作業中メールボックスのすべてのメッセージを返します。

- *searchCriteria*が複数の検索キーを含む場合、AND検索が行われます。
  searchCriteria = FLAGGED FROM "SMITH"
  は¥Flaggedフラグが付けられ、かつSmithから送信されたメッセージを返します。

- OR や NOT 演算子を使用できます:
  searchCriteria = OR SEEN FLAGGED
  は、¥Seenまたは¥Flaggedフラグが設定されたメッセージを返します。

  searchCriteria = NOT SEEN
  は、¥Seenフラグが設定されていないメッセージが返されます。

  searchCriteria = HEADER CONTENT-TYPE "MIXED" NOT HEADER CONTENT-TYPE "TEXT"
  は、content-typeヘッダに“Mixed”を含み、かつ“Text”を含まないメッセージが返されます。

  searchCriteria = HEADER CONTENT-TYPE "E" NOT SUBJECT "o" NOT HEADER CONTENT-TYPE "MIXED"
  は、content-typeヘッダに“ e ”を含み、かつSubjectヘッダに“ o ”を含まない、かつcontent-typeヘッダに“ Mixed ”を含まないメッセージが返されます。

  searchCriteria = OR (ANSWERED SMALLER 400) (HEADER CONTENT-TYPE "E" NOT SUBJECT "o" NOT HEADER CONTENT-TYPE "MIXED")
  最初の括弧内条件式に合致するか、または二番目の括弧内条件式に合致するメッセージが返されます。

  searchCriteria = OR ANSWERED SMALLER 400 (HEADER CONTENT-TYPE "E" NOT SUBJECT "o" NOT HEADER

CONTENT-TYPE "MIXED")
¥Answeredフラグが設定されている、またはメッセージ全体のサイズが400 byte未満で、かつ括弧内の条件式に合致するメッセージが返されます。

最後の二つの例題で見たとおり、括弧の有無で検索結果は異なるものとなります。

- *searchCriteria*にはオプションで*[CHARSET]*定義を含めることができます。これは "CHARSET" 文字列と登録済みの *[CHARSET] (US ASCII, ISO-8859)* からなります。これは*searchCriteria*文字列の文字コードを指定するものです。これを使用する場合、開発者は*searchCriteria* 文字列を指定した文字セットに変換しなければなりません。(4Dの **Mac to ISO** コマンドを参照してください。)
  デフォルトで、検索条件に拡張文字が含まれる場合、4D Internet Commandsは*searchCriteria* 文字列をQuotable Printableでエンコードします。

  searchCriteria = CHARSET "ISO-8859" BODY "Help"
  これは検索条件が文字セット iso-8859 を使用していることを意味します。サーバは必要に応じて検索条件を変換します。

検索値のタイプ

検索キーは、検索条件値を要求することがあります:

- **日付値による検索キー**

<日付> 文字列は以下の文字列で記述されます: date-day+"-"+date-month+"-"+date-year。date-dayは日を表す数字で最大2文字です。date-month は月の名前で (Jan/Feb/Mar/Apr/May/Jun/Jul/Aug/Sep/Oct/Dec) 、date-yearは年 (最大4文字)です。

例: `searchCriteria = SENTBEFORE 1-Feb-2000` (日付は特別文字を含まないのでクォートで囲む必要はありません。)

- **文字列による検索キー**

<文字列> は拡張文字を含む場合、クォートで囲まなければなりません。文字列がスペースなどの特別文字をまったく含まない場合は、クォートで囲まなくても構いません。文字列をクォートで囲むことで、その文字列が正しく解釈されることを期待できます。

例: `searchCriteria = FROM "SMITH"`

Note: 文字列を使用するすべての検索キーは、含む検索が行われます。検索は大文字小文字を区別しません。

- **フィールド名による検索キー**

<フィールド名> はヘッダフィールドの名前です。

例: `searchCriteria = HEADER CONTENT-TYPE "MIXED"`

- **フラグ値による検索キー**

<フラグ> は一つ以上の (標準フラグを含む) キーワードを受け入れ、スペースで区切られます。

例: `searchCriteria = KEYWORD \Flagged \Draft`

- **メッセージの組による検索キー**

複数のメッセージを指定します。メールボックス中の1からメッセージ総数までの連続したメッセージシーケンス番号から指定します。
コンマで個別の番号を区切り、コロンは二つの数字間の範囲を表します。

例:
2,4:7,9,12:* は、15のメッセージがある時、 2,4,5,6,7,9,12,13,14,15を示します。

`searchCriteria = 1:5 ANSWERED` はメッセージシーケンス番号が1から5までのメッセージで、**¥Answered**フラグが設定されたメッセージを検索します。

`searchCriteria= 2,4 ANSWERED` はメッセージシーケンス番号が2と4のメッセージで、**¥Answered**フラグが設定されたメッセージを検索します。

使用可能な検索キー

ALL: メールボックス中すべてのメッセージ
ANSWERED: **¥Answered**フラグが設定されたメッセージ
UNANSWERED: **¥Answered**フラグが設定されていないメッセージ
DELETED: **¥Deletes**フラグが設定されたメッセージ
UNDELETED: **¥Deletes**フラグが設定されていないメッセージ
DRAFT: **¥Draft**フラグが設定されたメッセージ
UNDRAFT: **¥Draft**フラグが設定されていないメッセージ
FLAGGED: **¥Flagged**フラグが設定されたメッセージ
UNFLAGGED: **¥Flagged**フラグが設定されていないメッセージ
RECENT: **¥Recent**フラグが設定されたメッセージ
OLD: **¥Recent**フラグが設定されていないメッセージ
SEEN: **¥Seen**フラグが設定されたメッセージ
UNSEEN: **¥Seen**フラグが設定されていないメッセージ
NEW: **¥Recent**フラグが設定されていて、¥Seenフラグが設定されていないメッセージ: これは"(RECENT UNSEEN)"と同じ意味です。
KEYWORD <フラグ>: 指定したキーワードが設定されたメッセージ
UNKEYWORD <フラグ>: 指定したキーワードが設定されていないメッセージ
BEFORE <日付>: 内部日付が指定した日付より前のメッセージ
ON <日付>: 内部日付が指定した日付のメッセージ
SINCE <日付>: 内部日付が指定した日付以降のメッセージ
SENTBEFORE <日付>: Dateヘッダが指定した日付より前のメッセージ
SENTON <日付>: Dateヘッダが指定した日付のメッセージ
SENTSINCE <日付>: Dateヘッダが指定した日付以降のメッセージ
TO <文字列>: TOヘッダに指定した文字列が含まれるメッセージ
FROM <文字列>: FROMヘッダに指定した文字列が含まれるメッセージ
CC <文字列>: CCヘッダに指定した文字列が含まれるメッセージ
BCC <文字列>: BCCヘッダに指定した文字列が含まれるメッセージ
SUBJECT <文字列>: Subjectヘッダに指定した文字列が含まれるメッセージ
BODY <文字列>: ボディにヘッダに指定した文字列が含まれるメッセージ
TEXT <文字列>: ヘッダまたはボディにヘッダに、指定した文字列が含まれるメッセージ
HEADER <フィールド名> <文字列>: 指定したヘッダが含まれかつ、フィールドの値に指定した文字列が含まれるメッセージ
UID <メッセージUID>: 指定したユニークIDに対応するメッセージ
LARGER <n>: メッセージサイズが指定したbyteより大きなメッセージ
SMALLER <n>: メッセージサイズが指定したbyteより小さなメッセージ
NOT <検索キー>: 指定した検索キーに合致しないメッセージ
OR <search-key1> <search-key2>: どちらかの検索キーに合致するメッセージ

IMAP_SetCurrentMB

IMAP_SetCurrentMB (imap_ID ; mbName ; msgNber ; newMsgNber ; customFlags ; permanentFlags ; mbUID) -> 戻り値

引数	型		説明
imap_ID	倍長整数	→	IMAPログイン参照
mbName	テキスト	→	選択するメールボックス名
msgNber	倍長整数	←	カレントメールボックスのメッセージ数
newMsgNber	倍長整数	←	¥Recentフラグがセットされたメッセージ数
customFlags	テキスト	←	メールボックスで現在使用されているフラグのリスト
permanentFlags	テキスト	←	恒久的に更新可能なフラグのリスト
mbUID	倍長整数	←	メールボックスユニークID
戻り値	整数	↩	エラーコード

説明

IMAP_SetCurrentMB コマンドは指定したメールボックスのメッセージを管理するために、セッションを開きます (例えばカレントメールボックスを選択します)。

接続中、一度に一つだけセッションを開くことができます。同時に複数のメールボックスにアクセスするには、複数の接続 (複数の *IMAP_Login*) が必要です。*IMAP_SetCurrentMB* コマンドは新しくメールボックスを選択する前に、自動でカレントのセッションを閉じます。このため、カレントのメールボックスが存在する状態で*IMAP_SetCurrentMB* コマンドの実行に失敗すると、その時点でカレントのメールボックスは選択されていないことになります。

新しいセッションを選択せずにセッションを閉じる (カレントメールボックスを閉じる) ことができます。これを行うには、*IMAP_SetCurrentMB* コマンドを存在しないメールボックス名で呼び出して返されるエラーを処理するか、*IMAP_CloseCurrentMB*または*IMAP_Logout* コマンドを実行します。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

mbName はカレントに指定する、存在するメールボックスの完全名です。

msgNber にはカレントメールボックスのメッセージ数が返されます (*IMAP_SetCurrentMB* が呼び出されるとき0に設定され、エラーの際には-1が返されます)。

newMsgNber にはカレントメールボックスの新規メッセージ数が返されます (*IMAP_SetCurrentMB* が呼び出されるとき0に設定され、エラーの際には-1が返されます)。

customFlags には、カレントメールボックスで使用されているフラグの完全なリストが返されます。*permanentFlags* にリストされているフラグのみが更新可能であることに注意してください。

permanentFlags には、恒久的に変更可能なメールボックスメッセージのフラグリストが返されます (IMAPサーバが管理する**¥Recent**フラグを除く)。(*IMAP_SetCurrentMB*が呼び出された時に空の文字列に設定されます。) *permanentFlags* 文字列には特別なフラグ **¥*** が含まれることがあります。このフラグは、メールボックスにこれらのフラグを格納しようとしたときにキーワードを作成することができることを意味します (*IMAP_SetFlags*参照)。

permanentFlags に空の文字列が返された場合、*customFlags* 引数にリストされたすべてのフラグを恒久的に変更可能であることを意味します。

mbUID にはカレントメールボックスのユニークIDが返されます。
このIDは特に、メールボックスが削除された後に同じ名前でメールボックスが再作成された場合に有効です。名前が同じなため、ユニークIDを使用しなければ、これが異なるメールボックスであることには気がつかないかもしれません。

IMAP_SetFlags

IMAP_SetFlags (imap_ID ; startMsg ; endMsg ; msgFlagsList ; deleteOption) -> 戻り値

引数	型		説明
imap_ID	倍長整数	→	IMAPログイン参照
startMsg	倍長整数	→	開始メッセージ番号
endMsg	倍長整数	→	終了メッセージ番号
msgFlagsList	文字	→	追加または削除するフラグ
deleteOption	整数	→	1 = フラグを追加, 0 = フラグを削除
戻り値	整数	←	エラーコード

説明

IMAP_SetFlags コマンドを使用して、指定した範囲のメッセージに対して一度に複数のフラグを設定、あるいは削除できます。

IMAPプロトコルではメッセージにフラグリストを付加することができます。**恒久タイプ**および**セッションのみ**、二つのタイプのフラグがあります。

恒久タイプのフラグはメッセージフラグから恒久的に追加または削除されます。 (*IMAP_SetCurrentMB*参照)。言い換えれば、**恒久タイプ**のフラグの変更は、セッションをまたいでも有効です。

セッションのみのフラグへの変更は、そのセッションでのみ有効です。

現在定義されているシステムフラグは:

- **Seen**: メッセージは既読である。
- **Answered**: メッセージは返信済みである。
- **Flagged**: メッセージは緊急/特別な注意が必要。
- **Deleted**: メッセージは*IMAP_Delete*, *IMAP_CloseCurrentMB*, *IMAP_SetCurrentMB* または *IMAP_Logout*により削除される。
- **Draft**: メッセージは完成しておらず、下書きである。
- **Recent**: メッセージはこのメールボックスに最近到着した。このセッションがこのメールが到着してから開かれる初めてのセッションである。以降のセッションではこのメッセージに対して**¥Recent**フラグを見ることはない。この恒久フラグはIMAPサーバにより管理され、*IMAP_SetFlags*を使用するなどしてIMAPクライアントから操作することはできない。

IMAPサーバは、クライアントが新しいフラグを定義することを許可することがあります。またIMAPサーバによっては上に示されたフラグ以外のフラグをサポートすることがあります。これはIMAPサーバの実装によります。この場合、これらの特別なフラグは“keywords”と呼ばれ、“¥”では始まりません (*IMAP_SetCurrentMB*参照)。

Note: **¥Deleted**フラグをセットして、*IMAP_SetCurrentMB*, *IMAP_CloseCurrentMB*, *IMAP_Delete* または *IMAP_Logout*を使用してカレントのセッションを閉じると、メッセージは恒久的に削除されます。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

startMsg は情報を取得するメッセージリストの開始メッセージ番号です。メッセージ番号は現在処理対象のメールボックス中のすべてのメッセージにおける、メッセージの位置を表します。

endMsg は情報を取得するメッセージリストの終了メッセージ番号です。メッセージ番号は現在処理対象のメールボックス中のすべてのメッセージにおける、メッセージの位置を表します。

Note: *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* そして*IMAP_CopyToMB*コマンドは、*startMsg*が*endMsg*よりも大きい場合でもエラーを返しません。この場合、コマンドは何も行いません。

msgFlagsList には一つ以上のフラグを含みます。複数のフラグを渡す場合、それぞれのフラグはスペースで区切ります。以下の例題を参照してください。

*permanentFlags*としてリストされたフラグのみが適用されます (*IMAP_SetCurrentMB*参照)。

deleteOption は、*msgFlagsList* 引数で指定したフラグを追加するか削除するかを指定する整数値です。

- 0を指定すると、*msgFlagsList*で指定されたフラグが取り除かれます。
- 1を指定すると、*msgFlagsList*で指定されたフラグが追加されます。

例題 1

フラグが既に付けられているかいないかにかかわらず、*startMsg*と*endMsg*で指定したメッセージに、**¥Answered**と**¥Draft**フラグを設定します:

```
msgFlagsName:="\Answered \Draft"
  ` \Answeredと\Draftはスペースで区切る
IMAP_SetFlags(imap_ID;startMsg;endMsg;msgFlagsName;1)
```

例題 2

フラグが既に付けられているかいないかにかかわらず、*startMsg*と*endMsg*で指定したメッセージから**¥Deleted**フラグを削除します:

```
msgFlagsName:="\Deleted"
IMAP_SetFlags(imap_ID;startMsg;endMsg;msgFlagsName;0)
```

例題 3

フラグが既に付けられているかいないかにかかわらず、*startMsg*と*endMsg*で指定したメッセージに**¥Deleted**フラグを設定します:

```
msgFlagsName:="\Deleted"
IMAP_SetFlags(imap_ID;startMsg;endMsg;msgFlagsName;1)
IMAP_CloseCurrentMB(imap_ID)
  `カレントのメールボックスを閉じ、指定したメッセージを恒久的に削除します
```

例題 4

CheckBoxAnswered チェックボックスの値に基づき、**¥Answered**フラグを設定します:

```
$Error:=IMAP_SetFlags(vImap_ID;$msgNum;$msgNum;"\Answered";Num(CheckBoxAnswered=0))
```

IMAP_SetPrefs

IMAP_SetPrefs (stripLineFeed ; msgFolder) -> 戻り値

引数	型		説明
stripLineFeed	整数	⇒	0 = ラインフィードを保持する, 1 = ラインフィードを取り除く, -1 = 変更しない
msgFolder	テキスト	⇒	メッセージフォルダパス ("" = 変更しない)
戻り値	整数	⇐	エラーコード

説明

*IMAP_SetPrefs*コマンドはIMAPコマンドの環境設定を行います。

stripLineFeed は保存されるメッセージでラインフィードを処理する方法を指定します。ほとんどのIMAPサーバは、行の終端としてキャリッジリターンとラインフィードの組を使用します。Macintoshアプリケーションは行の終端としてキャリッジリターンのみを使用します。このオプションはメッセージテキストからラインフィードを取り除くかどうかを指定します。0を渡すと、受信したメッセージをIMAPサーバに格納されたフォーマットそのままにします。1を渡すと、受信したメッセージからラインフィードを取り除きます。-1を渡すと以前に環境設定に設定した値そのままにします。デフォルトの設定値は1で、ラインフィードを自動で取り除きます。

*msgFolder*は、*IMAP_Download*コマンドで受信したメッセージの格納先フォルダを指定するローカルパス名です。

IMAP_SubscribeMB

IMAP_SubscribeMB (imap_ID ; mbName ; mbSubscribe) -> 戻り値

引数	型		説明
imap_ID	倍長整数	→	IMAPログイン参照
mbName	テキスト	→	購読開始または停止するメールボックス名
mbSubscribe	整数	→	0 = 購読しない; 1= 購読する
戻り値	整数	←	エラーコード

説明

IMAP_SubscribeMB コマンドを使用して、IMAPサーバーの"購読済み"ユーザーメールボックスに指定したメールボックス名を追加したり、またはそこから取り除くことができます。

このコマンドを利用することで、ユーザは数多くの利用可能なメールボックスの中から、通常閲覧を行うメールボックスを指定し、リストの数を減らすことができます。これを行うには、単に*IMAP_ListMBs* コマンドを*subscribedMB* オプション引数に1を指定して実行します (*IMAP_ListMBs*参照)。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

mbName は購読を開始または停止するメールボックスのフル名です。

mbSubscribe に0を渡すと購読を停止します。1を渡すと購読を開始します。

IMAP_UIDToMsgNum

IMAP_UIDToMsgNum (imap_ID ; unique_ID ; msgNum) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
unique_ID	倍長整数	⇒	メッセージユニークID値
msgNum	倍長整数	⇐	メッセージ番号
戻り値	整数	⇒	エラーコード

説明

IMAP_UIDToMsgNum コマンドは、メッセージの*unique_ID*値を、*imap_ID*で参照されるカレントのメールボックス中のメッセージリストの現在の*msgNum*に変換します。特定のメッセージのメッセージ番号は、メールリストの他の項目に相対的であるため、このコマンドはメッセージの現在の位置を取得するために使用できます。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

unique_ID にはメッセージのユニークIDを指定します。

msgNum には、*unique_ID*で指定された項目の現在のメッセージ番号 (現在のメッセージリスト中の位置) が返されます。サーバ上で*unique_ID*が見つからない場合、*msgNum*に0が返され、エラーは生成されません。

IMAP_VerifyID

IMAP_VerifyID (imap_ID) -> 戻り値

引数	型		説明
imap_ID	倍長整数	⇒	IMAPログイン参照
		⇐	0 = 接続は既に閉じられている
戻り値	整数	⇒	エラーコード

説明

IMAPサーバは、管理者が設定した時間アクションがない場合、自動で接続を閉じます。IMAPサーバに対して動作するそれぞれのコマンドはこのタイマをリセットします。*IMAP_VerifyID* コマンドは指定されたIMAP接続のタイマを、他のアクションを起こさずに、リセットします。これによりタイムアウトさせずに接続を保持することができます。

IMAP_VerifyID コマンドを実行すると、コマンドは接続が既に閉じられていないかチェックします。接続がまだ開かれていれば、このコマンドはIMAPサーバのタイマをリセットします。接続が閉じられていれば、*IMAP_VerifyID* は適切なエラーを返し、IMAP接続に使用されていたメモリを解放したうえで、*imap_ID*に0を返します。

imap_ID は*IMAP_Login*で作成されるIMAPログイン参照です。

IC POP3 メール閲覧

- メール受信 - 概要
- POP3_BoxInfo
- POP3_Charset
- POP3_Delete
- POP3_DownLoad
- POP3_GetMessage
- POP3_GetPrefs
- POP3_Login
- POP3_Logout
- POP3_MsgInfo
- POP3_MsgLst
- POP3_MsgLstInfo
- POP3_Reset
- POP3_SetPrefs
- POP3_UIDToNum
- POP3_VerifyID

メール受信 - 概要

一連のPOP3コマンドを使用して、POP3メールサーバからメールの受信が可能になります。4D Internet CommandsはMIME互換であり、複数のパートからなるメッセージを認識および展開できます。

POP3関連のコマンドは二つのセクション"**IC POP3 メール閲覧**"と"**IC ダウンロードしたメール**"に分けられます。この分離はメールを読み込む異なる方法を表しています。POP3サーバからメールを読みだす際、メッセージ (またはメッセージに関する情報) は (変数, フィールド, 配列など) 4Dのオブジェクトに格納されるか、ディスクにダウンロードされます。この"**IC POP3 メール閲覧**"セクションでは、4D Internet Commandsを使用してPOP3サーバから4Dオブジェクトにメッセージを読み込むコマンドを説明します。

メッセージの受信に二つの方法が用意されている理由は、何メガバイトもあるメッセージをダウンロードするかもしれない場合のメモリの制限のためです。例えば、5MBの添付ファイルがあるメッセージは、容易にデータベースの格納容量をオーバーフローさせます。4Dオブジェクトでこのサイズを格納できるのはピクチャやBlobです。しかしメッセージや添付ファイルをこれらのオブジェクトに変換するのは、大きなメモリ空間が必要なため、効率的ではありません。これを解決するために、このセクションには*POP3_Download*コマンドが用意され、POP3サーバからメッセージをローカルディスクにダウンロードできるようになっています。ディスクに保存された後は、"**IC ダウンロードしたメール**"セクションのコマンドを使用して、ファイルを操作できます。

一連のPOP3コマンドを使用する際、よく使用される引数、特に*msgNumber*と*uniqueID*について理解することが重要です。*msgNumber*は、*POP3_Login*が実行された際のメールボックス内のメッセージ毎の番号です。ログイン時に、メールボックス内のメッセージには1からメッセージ数までの数値が割り当てられます。番号はメールボックスに届いた順に付けられ、1がもっとも古いメッセージとなります。この番号は*POP3_Login*から*POP3_Logout*までの間のみ有効です。

*POP3_Logout*が実行された時、削除マークが付けられたメッセージは削除されます。ユーザが再度サーバにログインすると、現在メールボックスにあるメッセージに再び番号が割り当てられます。例えば、メールボックスに10のメッセージがあり、1から5までのメッセージが削除されると、次回のログイン時には6から10番目のメッセージが1から5に番号付けされます。

POP3サーバにログインして、以下のメッセージリストを所得したとします。

#	UniqueID	日付	送信者	件名
1	bd573a4dbd573a4d	1 Jul 1998 …	jimw@acme.com	Sales lead…
2	bd574dc7bd574dc7	1 Jul 1998 …	frank@acme.com	Site-License order
3	bd575f06bd575f06	3 Jul 1998 …	joe@acme.com	Lunch anyone?
4	bd5761d4bd5761d4	4 Jul 1998 …	kelly@acme.com	Your wife called…
5	bd577dc7db577dc5	4 Jul 1995 …	track@fedex.com	FedEx tracking

セッション中メッセージ3と4を削除します。セッションをログアウトすると、リクエスト済みの削除が実行されます。再度ログインするとメッセージリストは以下のようになります:

#	UniqueID	日付	送信者	件名
1	bd573a4dbd573a4d	1 Jul 1998 …	jimw@acme.com	Sales lead…
2	bd574dc7bd574dc7	1 Jul 1998 …	frank@acme.com	Site-License order
3	bd577dc7db577dc5	4 Jul 1995 …	track@fedex.com	FedEx tracking

*msgNumber*はスタティックな値ではなく、セッションが開かれた時のメールボックス中のメッセージに基づいて割り当てられます。ところで、*uniqueID*はユニークな値で、サーバがメッセージを受信した際に割り当てられます。このIDはPOP3サーバが、メッセージを受信したときの日付や時間に基づいて計算し、割り当てます。残念ながらPOP3サーバは*uniqueID*をメッセージの主たる参照としては扱いません。POP3コマンドを通じて、サーバのメッセージを参照するためには*msgNumber*を使用します。データベースにメッセージ参照を格納し、メッセージはサーバに残すようなソリューションを構築する際、開発者はこの点に留意する必要があります。

Note: 4D Internet commandsではPOP3, IMAP, FTP接続参照を直接低レベルTCPコマンドに渡すことができ、またその逆も可能です。詳細は **低レベルルーチン - 概要**セクションを参照してください。

POP3_BoxInfo

POP3_BoxInfo (pop3_ID ; msgCount ; msgSize) -> 戻り値

引数	型		説明
pop3_ID	倍長整数	⇒	POP3ログイン参照
msgCount	倍長整数	⇐	メッセージ数
msgSize	倍長整数	⇐	メッセージの総サイズ
戻り値	整数	⇒	エラーコード

説明

POP3_BoxInfo コマンドは、*pop3_ID*で参照されるセッションの、メールボックスの現時点でのメッセージ数やサイズなどの情報を返します。

pop3_ID は*POP3_Login*で作成される、開かれたセッションの参照です。

msgCount にはメールボックス中のメッセージ数が返されます。

msgSize にはメールボックス中のメッセージの合計サイズが返されます。

POP3_Charset

POP3_Charset (decodeHeaders ; bodyCharset) -> 戻り値

引数	型		説明
decodeHeaders	整数	➡	-1 = 現在の設定を使用, 0 = 管理しない, 1 = ISO-8859-1またはISO-2022-JPの場合Mac OS文字セットに変換、拡張文字をデコード
bodyCharset	整数	➡	-1 = 現在の設定を使用, 0 = 管理しない, 1 = ISO-8859-1またはISO-2022-JPの場合Mac OS文字セットに変換
戻り値	整数	⮌	エラーコード

説明

POP3_Charset コマンドはPOP3やMSGコマンドでメッセージを処理する際の、拡張文字の自動サポートを設定します。このコマンドが呼び出されないか引数に0が設定されていると、バージョン6.7以降の4D Internet Commandsもバージョン6.5.xと同様に動作します。

POP3_Charset を使用してまずヘッダーの拡張文字のデコード管理方法を指定し、次にメッセージボディとヘッダーの文字セットの変換の管理を指定します。

このコマンドは、“Subject” やメールアドレスなどのメッセージヘッダーに含まれる拡張文字のサポートに有用です (例えば“=?ISO-8859-1?Q?Test=E9?= <test@n.net >”のようなメールアドレスのデコード)。

decodeHeaders 引数は、*POP3_MsgLst* や *MSG_FindHeader* (互換性の注意を参照) コマンドが実行される際に、ヘッダーのデコードや変換の処理方法を指定します。デフォルト値は0です。

- -1: 現在の設定を使用
- 0: 管理しない
- 1: 必要に応じてヘッダをデコードする。デコードが行われまた指定された文字セットがISO-8859-1またはISO-2022-JPである場合、ヘッダーは4D文字列に自動で変換されます。

互換性の注意 (version 6.8.1): *MSG_Charset* コマンドが事前に実行されていなければ、*POP3_Charset*は*POP3_MsgLst*と同様に*MSG_FindHeader*にも適用されます。

bodyCharset 引数は、*MSG_GetBody* (互換性の注意を参照) コマンドが実行される際に、メッセージボディの文字セットの変換方法を指定します。デフォルト値は0です。

- -1: 現在の設定を使用
- 0: 管理しない
- 1: “Body-Content-Type”の文字セットがISO-8859-1またはISO-2022-JPに設定されている場合、メッセージボディは4D文字列に自動で変換されます。

互換性の注意 (version 6.8.1): *MSG_Charset* コマンドが事前に実行されていなければ、*POP3_Charset*は*MSG_GetBody* にも適用されます。

例題 1

バージョン6.5.x の4D Internet Commandsを使用する:

```
$Err:=MSG_FindHeader($msgfile;"From";$from)
$from:=ISO to Mac($from)
$Err:=MSG_FindHeader($msgfile;"To";$to)
$to:=ISO to Mac($to)
$Err:=MSG_FindHeader($msgfile;"Cc";$cc)
$cc:=ISO to Mac($cc)
```

```
$Err:=MSG_FindHeader($msgfile;"Subject";$subject)
$subject:=ISO to Mac($subject)

$Err:=MSG_MessageSize($msgfile;$HdrSize;$BdySize;$MsgSize)
$Err:=MSG_GetBody($msgfile;0;$BdySize;$BodyContent)
$BodyContent:=ISO to Mac($BodyContent)
```

例題 2

バージョン6.7の4D Internet Commandsを使用する:

```
$Err:=POP3_Charset(1;1)
$Err:=MSG_FindHeader($msgfile;"From";$from)
$Err:=MSG_FindHeader($msgfile;"To";$to)
$Err:=MSG_FindHeader($msgfile;"Cc";$cc)
$Err:=MSG_FindHeader($msgfile;"Subject";$subject)

$Err:=MSG_MessageSize($msgfile;$HdrSize;$BdySize;$MsgSize)
$Err:=MSG_GetBody($msgfile;0;$BdySize;$BodyContent)
```

例題 3

バージョン6.8の4D Internet Commandsを使用する:

```
$Err:=MSG_Charset(1;1)
$Err:=MSG_FindHeader($msgfile;"From";$from)
$Err:=MSG_FindHeader($msgfile;"To";$to)
$Err:=MSG_FindHeader($msgfile;"Cc";$cc)
$Err:=MSG_FindHeader($msgfile;"Subject";$subject)

$Err:=MSG_MessageSize($msgfile;$HdrSize;$BdySize;$MsgSize)
$Err:=MSG_GetBody($msgfile;0;$BdySize;$BodyContent)
```

POP3_Delete

POP3_Delete (pop3_ID ; startMsg ; endMsg) -> 戻り値

引数	型		説明
pop3_ID	倍長整数	→	POP3ログイン参照
startMsg	倍長整数	→	開始メッセージ番号
endMsg	倍長整数	→	終了メッセージ番号
戻り値	整数	←	エラーコード

説明

POP3_Delete コマンドは、*startMsg*から*endMsg*で指定した範囲のメッセージに削除フラグを立てます。削除動作は*POP3_Logout* コマンドを発行するまで行われません。*POP3_Logout* コマンドを実行する前に、何らかの理由 (タイムアウト、ネットワークの問題など) でカレントセッションが終了した場合は、削除フラグがたてられたメッセージはPOP3サーバに残ります。

pop3_ID は*POP3_Login*で作成される、開かれたセッションの参照です。

startMsg は削除するメッセージの開始位置を指定するメッセージ番号です。

endMsg は削除するメッセージの終了位置を指定するメッセージ番号です。

Note: *POP3_Delete*, *POP3_MsgLstInfo* および *POP3_MsgLst* コマンドは、*startMsg*が*endMsg*より大きくてもエラーを返しません。このような場合、これらのコマンドは何も行いません。

POP3_DownLoad

POP3_DownLoad (pop3_ID ; msgNumber ; headerOnly ; fileName) -> 戻り値

引数	型		説明
pop3_ID	倍長整数	➡	POP3ログイン参照
msgNumber	倍長整数	➡	メッセージ番号
headerOnly	整数	➡	0 = メッセージ全体, 1 = ヘッダのみ
fileName	テキスト	➡	ローカルファイル名
		⬅	結果のローカルファイル名
戻り値	整数	↩	エラーコード

説明

POP3_DownLoad コマンドは、POP3サーバからメッセージをディスク上のファイルにダウンロードするために使用します。添付ファイルを含むまたは32Kサイズを超えるすべてのPOP3メッセージは、このコマンドを使用してダウンロードすべきです。メッセージに添付されたファイルはこの方法でダウンロードされたファイルからのみ展開できます。

pop3_ID は*POP3_Login*で作成される、開かれたセッションの参照です。

msgNumber は取得したいメッセージのメールボックス中のメッセージ番号です。*msgNumber*は現在のメッセージリストの位置を表します。*msgNumber*はカレントセッションの中でのみ有効で、セッションが異なれば、そのメッセージ番号が指定するメッセージは前回と異なるかもしれません。

headerOnly はメッセージ全体を取得するか、ヘッダのみを取得するか指定するための整数値です。

fileName は、メッセージの保存先ファイル名とオプションのパスを渡します。この引数の指定方法は三つあります:

""	= *POP3_SetPrefs*で指定したフォルダに"temp1"という名前でメッセージを格納します。(同じ名前のファイルが既に存在する場合、使用されていないファイル名が見つかるまで、"temp2", "temp3"が試されます。)
"ファイル名"	= *POP3_SetPrefs* で指定したフォルダに、*fileName*でメッセージを格納します。
"パス:ファイル名"	= *fileName*で指定したパスにメッセージを格納します。

最初の二つの方法では、*POP3_SetPrefs*でフォルダが指定されていない場合、メッセージは (4Dシングルユーザの場合) データベースストラクチャと同じフォルダに、または (4D Serverの場合) 4Dクライアントフォルダに保存されます。ファイルがディスクに保存された後、ファイルが実際に保存された名称が*fileName* 引数に渡された変数に返されます。*fileName*に既に存在するファイル名を指定して*POP3_Download*コマンドを呼び出すと、増分された数値がファイル名に付加され、実際にディスクに保存されたファイル名が*fileName*変数に返されます。

POP3_GetMessage

POP3_GetMessage (pop3_ID ; msgNumber ; offset ; length ; msgText) -> 戻り値

引数	型		説明
pop3_ID	倍長整数	⇒	POP3ログイン参照
msgNumber	倍長整数	⇒	メッセージ番号
offset	倍長整数	⇒	受信を開始する文字位置オフセット
length	倍長整数	⇒	受信する文字数
msgText	テキスト	⇐	メッセージテキスト
戻り値	整数	⇐	エラーコード

説明

POP3_GetMessage コマンドは、*pop3_ID*で指定したメールボックス中、*msgNumber*で指定したメッセージの完全なテキストを返します。*POP3_SetPrefs* コマンドで特に指定されていなければ、メッセージ中のラインフィードは取り除かれます。*POP3_GetMessage* コマンドはヘッダ情報を含む、メッセージの全ブロックを返します。

pop3_ID は*POP3_Login*で作成される、開かれたセッションの参照です。

msgNumber は取得したいメッセージのメールボックス中のメッセージ番号です。*msgNumber*は現在のメッセージリストの位置を表します。*msgNumber*はカレントセッションの中でのみ有効で、セッションが異なればそのメッセージ番号が指定するメッセージは前回と異なるかもしれません。

offset は読み込みを開始する文字位置を指定するための倍長整数値です。ほとんどの場合この引数には0を渡します。

length は*offset*から読み込むバイト数を指定する倍長整数値です。過去の経緯からこの引数の最大長は32,000バイトに制限されているため、*length* 引数も32,000以下にすべきです。このサイズを超えるメッセージは*POP3_Download* コマンドでディスクにダウンロードしなければなりません。

msgText は受信したメッセージを受け取るテキスト変数です。

POP3_GetPrefs

POP3_GetPrefs (stripLineFeed ; msgFolder ; attachFolder) -> 戻り値

引数	型		説明
stripLineFeed	整数	⇐	0 = ラインフィードを保持する, 1 = ラインフィードを取り除く
msgFolder	テキスト	⇐	メッセージフォルダパス ("" = 変更しない)
attachFolder	テキスト	⇐	添付フォルダパス ("" = 変更しない)
戻り値	整数	⇒	エラーコード

説明

POP3_GetPrefs コマンドはPOP3コマンド用の現在の環境設定値を返します。環境設定値は引数にリストされた変数に返されます。

stripLineFeed にはラインフィードの処理に関する設定値が返されます。

msgFolder には受信したメッセージの格納先フォルダのパスが返されます。

attachFolder には展開された添付ファイルの格納先フォルダのパスが返されます。

互換性の注意 (version 6.8.1): *POP3_GetPrefs* コマンドの*attachFolder* 引数はオプションです。この引数は使用されないため、渡さないことをお勧めします。この設定はMSGコマンドのためのものであり、POP3には関連がない点に注意してください。

POP3_Login

POP3_Login (hostName ; userName ; password ; aPOP ; pop3_ID {; sessionParam}) -> 戻り値

引数	型		説明
hostName	文字	⇒	POP3メールサーバのホスト名またはIPアドレス
userName	文字	⇒	ユーザ名
password	文字	⇒	パスワード
aPOP	整数	⇒	0 = クリアテキストログイン, 1 = APOP ログイン
pop3_ID	倍長整数	⇐	POP3ログイン参照
sessionParam	倍長整数	⇒	1 = SSLを使用, 0または省略 = SSLを使用しない
戻り値	整数	⇐	エラーコード

説明

POP3_Login コマンドは、*userName* と*password*を使用してPOP3メールサーバにログインします。*aPOP*に1を指定した場合、APOPメカニズム (RFC#1321) がログインに使用されます。*aPOP*が0の場合、通常のクリアテキストパスワードログインが実行されます。このコマンドの実行が成功すると、*pop3_ID*に参照が返されます。

警告： POP3サーバは、対話的な方法でアクセスされるようにはデザインされていません。サーバにログインした後は、必要なアクションを行い、できる限り早くサーバからログアウトすべきです。*POP3_Login*と*POP3_Logout*との間でユーザの指示を待ちうけるような画面を表示させるべきではありません。POP3サーバは特定の時間アクションがない接続を自動で切断します。 POP3のRFCによれば、このタイマは最低30分が想定されています。しかし経験上、ほとんどのサーバはもっと短い時間で接続を切断します。

POP3サーバと通信を行うそれぞれのコマンドは、このタイマをリセットします。*POP3_Logout*前にサーバが接続をアボートする場合、削除処理はロールバックされます。

hostName はPOP3メールサーバのホスト名またはIPアドレスです。ホスト名の利用をお勧めしますが、必要であればIPアドレスも利用できます。

userName はPOP3メールサーバにログインするためのユーザ名です。*userName* にはドメイン名を含めるべきではありません。例えば"jack@4d.com"というアドレスでは、*userName* はおそらく"jack"になります (設定により異なることがあります)。

password はPOP3メールサーバにログインするためのパスワードです。

aPOP は、ログインにAPOPメカニズムを使用するかどうか指定する整数値です。1を指定するとAPOPメカニズムが使用されます。0を渡すとクリアテキストパスワードログインが行われます。デフォルトは0です。

pop3_ID は倍長整数変数で、セッションの参照が返されます。このセッションのアクションを実行する際にこの参照を使用します。

オプションの*sessionParam*引数を使用すると、接続にSSLプロトコルを使用することができます:

- 1を渡すと、POP3サーバーへの接続はSSLで行われます (同期モード)、
- 0を渡すかこの引数を省略すると、接続は標準の非保護モードで行われます。

POP3_Logout

POP3_Logout (pop3_ID) -> 戻り値

引数	型		説明
pop3_ID	倍長整数	→	POP3ログイン参照
		←	0 = ログアウト成功
戻り値	整数	↩	エラーコード

説明

POP3_Logout コマンドは、*pop3_ID*で指定されたPOP3セッションからログアウトします。ログアウトに成功すると*pop3_ID*変数に0が返されます。

POP3サーバからログアウトすると、セッション中に削除フラグを立てたメッセージを削除するよう、サーバに指示することにもなります。削除を無効にするには、*POP3_Logout*の前に*POP3_Reset* コマンドを使用します。

pop3_ID は*POP3_Login*で作成される、開かれたセッションの参照です。

POP3_MsgInfo

POP3_MsgInfo (pop3_ID ; msgNumber ; msgSize ; uniqueID) -> 戻り値

引数	型		説明
pop3_ID	倍長整数	⇒	POP3ログイン参照
msgNumber	倍長整数	⇒	メッセージ番号
msgSize	倍長整数	⇐	メッセージサイズ
uniqueID	文字	⇐	サーバ上のメッセージのユニークID
戻り値	整数	↩	エラーコード

説明

POP3_MsgInfo コマンドは、*pop3_ID*で参照される開かれたメールボックス中、*msgNumber*で指定したメッセージの情報を返します。メッセージのサイズとユニークIDに関する情報が返されます。

pop3_ID は*POP3_Login*で作成される、開かれたセッションの参照です。

msgNumber は情報を取得したいメッセージのメールボックス中のメッセージ番号です。*msgNumber*は現在のメッセージリストの位置を表します。*msgNumber*はカレントセッションの中でのみ有効で、セッションが異なれば、そのメッセージ番号が指定するメッセージは前回と異なるかもしれません。

msgSize には*msgNumber*で指定したメッセージのサイズが返されます。

uniqueID にはサーバ上のメッセージのユニークIDが返されます。*uniqueID*はPOP3サーバアプリケーションによってメッセージに割り当てられる値です。この値は、*msgNumber*と異なり、セッションをまたがっても変更されることはありません。*uniqueID*は、データベースに既にメッセージをダウンロード済みかどうかを知るために利用できます。

POP3_MsgLst

POP3_MsgLst (pop3_ID ; start ; end ; hdrArray ; msgNumArray ; idArray ; valueArray) -> 戻り値

引数	型		説明
pop3_ID	倍長整数	➡	POP3ログイン参照
start	倍長整数	➡	開始メッセージ番号
end	倍長整数	➡	終了メッセージ番号
hdrArray	文字配列	➡	受信するヘッダの配列
msgNumArray	倍長整数配列	⬅	メッセージ番号配列
idArray	文字配列	⬅	ユニークID配列
valueArray	2D文字列配列, 2Dテキスト配列	⬅	ヘッダ値の二次元配列
戻り値	整数	↩	エラーコード

説明

POP3_MsgLst コマンドは、メールボックスの内容の、特定の情報を取得するために使用します。*hdrArray* は情報を取得したいヘッダを指定する文字列またはテキスト配列です。*valueArray*は二次元配列で、*hdrArray*で指定したヘッダの情報を受け取ります。リクエストされたそれぞれのヘッダは、*valueArray*の一次元目に対応する配列をもちます。

このコマンドを使用して、メッセージリストの特定の列をリクエストできます。このコマンドはヘッダ項目の値のみを返します。メッセージのボディを取得することはできません。

Note: メールヘッダには拡張文字が含まれている場合があるため、*POP3_Charset* コマンドを使用してそれらの処理を自動化することができます。

例題

```
aHeaders{1}:="Date:"
aHeaders{2}:="From:"
aHeaders{3}:="Subject:"
POP3_MsgLst(<>POP3_ID;vStart;vEnd;aHeaders;aMsgNum;aUIDs;aValues)
aValues{1}{1}may equal"Thu, 19 November 1998 00:24:02 -0800"
aValues{2}{1}may equal"Jack@4d.com"
aValues{3}{1}may equal"Call your wife"
```

エラーは以下のように処理されます:

1. 通信に関連するエラーのみが返されます。コマンドがネットワークやシンタックス、サーバなどのエラーのため正しく実行できなかった場合、対応するエラーコードが返されます。
2. 指定された範囲のメッセージが存在しなかった場合やエラーを受け取った場合、
   - そのメッセージの配列要素は作成されません。
   - エラーコードは返されません。
3. 指定したヘッダが一部あるいはすべて、メッセージ中で全く見つからなかった場合、エラーは生成されません。
   - メッセージの配列要素は作成されます。
   - メッセージ番号とユニークID配列の要素には正しい値が返されます。
   - メッセージに存在しないヘッダに対しては、配列要素に空の文字列が返されます。
   - エラーコードは返されません。

Note: *POP3_Delete*, *POP3_MsgLstInfo* および *POP3_MsgLst* コマンドは、*startMsg*が*endMsg*より大きくてもエラーを返しません。このような場合、これらのコマンドは何も行いません。

POP3_MsgLstInfo

POP3_MsgLstInfo (pop3_ID ; startMsg ; endMsg ; sizeArray ; msgNumArray ; idArray) -> 戻り値

引数	型		説明
pop3_ID	倍長整数	→	POP3ログイン参照
startMsg	倍長整数	→	開始メッセージ番号
endMsg	倍長整数	→	終了メッセージ番号
sizeArray	倍長整数配列	←	サイズ配列
msgNumArray	倍長整数配列	←	メッセージ番号配列
idArray	文字配列	←	ユニークID配列
戻り値	整数	↩	エラーコード

説明

POP3_MsgLstInfo コマンドは、メールボックス中のメッセージの情報を返します。情報は三つの配列に返され、配列のそれぞれの要素が一つのメッセージに対応します。それぞれのメッセージサイズ、メッセージ番号、ユニークIDが返されます。配列は事前に定義されていなければなりません。*POP3_MsgLstInfo* コマンドは配列のサイズを受信したメッセージ数に設定します。

POP3_MsgLstInfo コマンドは、現在のメッセージリストの情報取得に失敗してもエラーを返しません。エラーが発生した場合、配列要素が作成されません。コマンドがそれぞれのメッセージを正しく読み込むと、*msgNumArray*には番号が順番通りに格納されます。問題が発生すると、*msgNumArray*に格納された数値に抜けが生じます。

pop3_ID は*POP3_Login*で作成される、開かれたセッションの参照です。

startMsg は、情報を取得するメッセージの開始位置を指定するメッセージ番号です。メッセージ番号は、*pop3_ID*で特定されるメールボックスのすべてのメッセージリスト中で位置を表す数値です。

endMsg は、情報を取得するメッセージの終了位置を指定するメッセージ番号です。メッセージ番号は、*pop3_ID*で特定されるメールボックスのすべてのメッセージリスト中で位置を表す数値です。

sizeArray は倍長整数配列で、*startMsg*と*endMsg*との間のそれぞれのメッセージサイズを受け取ります。

msgNumArray は倍長整数配列で、*startMsg*と*endMsg*との間のそれぞれのメッセージ番号を受け取ります。

idArray は文字またはテキスト配列で、*startMsg*と*endMsg*との間のそれぞれのメッセージのユニークIDを受け取ります。

Note: *POP3_Delete*, *POP3_MsgLstInfo* および *POP3_MsgLst* コマンドは、*startMsg*が*endMsg*より大きくてもエラーを返しません。このような場合、これらのコマンドは何も行いません。

POP3_Reset

POP3_Reset (pop3_ID) -> 戻り値

引数	型		説明
pop3_ID	倍長整数	⇒	POP3ログイン参照
戻り値	整数	⇐	エラーコード

説明

POP3_Reset コマンドはメッセージ数をリセットし、カレントセッションで削除フラグがたてられたメッセージの削除フラグを外します。

Note: *POP3_Delete* コマンドはメッセージに削除フラグを立てるだけです。POP3サーバ上のメッセージはログアウト (*POP3_Logout*) が成功したときにのみ削除されます。

pop3_ID は*POP3_Login*で作成される、開かれたセッションの参照です。

POP3_SetPrefs

POP3_SetPrefs (stripLineFeed ; msgFolder ; attachFolder) -> 戻り値

引数	型		説明
stripLineFeed	整数	→	0 = ラインフィードを保持する, 1 = ラインフィードを取り除く, -1 = 変更しない
msgFolder	テキスト	→	メッセージフォルダパス ("" = 変更しない)
attachFolder	テキスト	→	添付フォルダパス ("" = 変更しない)
戻り値	整数	←	エラーコード

説明

POP3_SetPrefs コマンドはPOP3コマンドの環境設定を行います。

stripLineFeed は保存されるメッセージでラインフィードを処理する方法を指定します。ほとんどのPOP3サーバは、行の終端としてキャリッジリターンとラインフィードの組を使用します。Macintoshアプリケーションは行の終端としてキャリッジリターンのみを使用します。このオプションはメッセージテキストからラインフィードを取り除くかどうかを指定します。0を渡すと、受信したメッセージをPOP3サーバに格納されたフォーマットそのままにします。1を渡すと、受信したメッセージからラインフィードを取り除きます。-1を渡すと以前に環境設定に設定した値そのままにします。デフォルトの設定値は1で、ラインフィードを自動で取り除きます。

*msgFolder*は、*POP3_Download* コマンドで受信したメッセージの格納先フォルダを指定するローカルパス名です。

互換性の注意 (version 6.8.1): *stripLineFeed*と*msgFolder*引数は、以前はMSG_系のコマンドに適用されていました。*MSG_SetPrefs* コマンドを使用した場合、このコマンドはMSG_系にはもう適用されません。

attachFolder は、*MSG_Extract* コマンドがメッセージボディから分離した添付ファイルの格納先フォルダを指定するローカルパス名です。

互換性の注意 (version 6.8.1): *attachFolder* 引数は*POP3_SetPrefs*と*MSG_SetPrefs*に存在します。どちらのコマンドを使用してもこの設定を変更できますが、*MSG_SetPrefs* コマンドの利用を強くお勧めします。*POP3_SetPrefs* 引数は互換性のために残されていて、将来は削除されます。*POP3_SetPrefs* コマンドの*attachFolder* 引数はオプションであり、この引数を使用しないことをお勧めします。同様のことが*POP3_GetPrefs*にもあてはまります。

POP3_UIDToNum

POP3_UIDToNum (pop3_ID ; uniqueID ; msgNumber) -> 戻り値

引数	型		説明
pop3_ID	倍長整数	⇒	POP3ログイン参照
uniqueID	文字	⇒	サーバ上のメッセージのユニークID
msgNumber	倍長整数	⇐	メッセージ番号
戻り値	整数	↩	エラーコード

説明

POP3_UIDToNum コマンドはメッセージのユニークIDを、*pop3_ID*で指定されるメールボックス中のメッセージリストの、現在の*msgNumber*に変換します。特定のメールメッセージの*msgNumber*はメールリスト中の他のメッセージに対し相対的です。このコマンドは現在のメッセージの位置情報を返します。

pop3_ID は*POP3_Login*で作成される、開かれたセッションの参照です。

uniqueID はPOP3サーバ上のメッセージを特定するために使用するユニークIDです。このコマンドは、*pop3_ID*で参照されるアカウントのメッセージヘッダからこの値を探します。見つかると、メッセージの現在の位置が*msgNumber*に返されます。

msgNumber には、*uniqueID*で指定されたメールの現在のメッセージ番号が返されます。サーバ上で*uniqueID*が見つからない場合、*msgNumber*に0が返され、エラーは生成されません。

POP3_VerifyID

POP3_VerifyID (pop3_ID) -> 戻り値

引数	型		説明
pop3_ID	倍長整数	⇒	POP3ログイン参照
		⇐	0 = 接続は既に閉じられている
戻り値	整数	⇒	エラーコード

説明

POP3サーバは、サーバ管理者によって指定された時間アクションがない接続については、自動で接続を閉じます。POP3サーバに対して動作するそれぞれのコマンドは、このタイマーをリセットします。*POP3_VerifyID* コマンドは、POP3セッションに対するアクションを行いませんが、このタイマーをリセットします。これにより、セッションがタイムアウトするかもしれない状況で、それが発生することを避けることができます。

POP3_VerifyID コマンドが実行されると、コマンドは接続が既に切断されていないか検証します。セッションがまだ開かれていれば、POP3サーバにタイマーをリセットするよう伝えます。セッションが既に閉じられている場合、*POP3_VerifyID* はエラーを返し、POP3セッションで使用されていたメモリを解放し、*pop3_ID*に0を返します。

pop3_ID は*POP3_Login*で作成される、開かれたセッションの参照です。

IC TCP/IP

- 低レベルルーチン - 概要
- TCP_Close
- TCP_Listen
- TCP_Open
- TCP_Receive
- TCP_ReceiveBLOB
- TCP_Send
- TCP_SendBLOB
- TCP_State

低レベルルーチン - 概要

Transmission Control Protocol/Internet Protocol (TCP/IP) は、インターネット上にデータを送信する際に使用される主なプロトコルです。4D Internet Commandsで提供されるTCPコマンドを使用すると、開発者はTCPセッションを確立し、TCPパケットを送受信できるようになります。

TCP接続を確立する方法は二つあります。ひとつは*TCP_Open* コマンドを実行することです。このコマンドは指定したドメインに、指定したポートで接続を開きます。*TCP_Open*では、SSL (Secured Socket Layer) を使用できます。もう一つの方法は*TCP_Listen* コマンドを実行することです。このコマンドは指定したドメインとポートを開き、やってくる接続を待ちます。接続が確立されたかを知る最適な方法は、*TCP_Listen* コマンド使用後に*TCP_State*コマンドでセッションの状態を確認することです。現在のセッションの状態を示すコードがこのコマンドからは返されます。その後、*TCP_Open*で確立した接続でできるのと同じように、TCPパケットの送受信ができます。

どのような場合でも、開かれたTCP接続は*TCP_Close* コマンドコマンドを使用して閉じなければなりません。

低レベルTCP/IPコマンドは通信のプロトコルに関する詳細な知識を必要とします。これらのルーチンを使用する開発者は、実装しようとするプロトコルについて完全な理解が必要です。さまざまなTCP/IPに割り当てられたポート番号、通信プロトコル、アドレッシングについての情報はRFCで見つけることができます。

TCPコマンドの接続参照

4D Internet commandsではPOP3, IMAP, FTP接続参照を直接低レベルTCPコマンドに渡したり、その逆ができます。

実際、プロトコルは絶えず進化するので、新しいコマンドが必要になることがあります。またいくつかのソフトウェアパッケージはRFCを独自に解釈し、標準的な実装が使用できなくなる場合もあります。既存のコマンドを使用する代わりに、あるいは存在しない機能を実装するために、低レベルTCPコマンドを使用して開発者は必要な高レベル関数を作成できます。

開発者はプロトコルが必要とするコマンドをすべて書き直さなくともよくなるため、互換性や開発の可能性が格段に向上します。

この例題では、*IMAP_Capability* コマンドと同機能の関数をTCP_IPコマンドで記述します。

- これは*IMAP_Capability* コマンドを使用した初期のメソッドです:

```
$ErrorNum:=IMAP_Login(vHost;vUserName;vUserPassword;vImap_ID)
If($ErrorNum=0)
   C_TEXT(vCapability)
   $ErrorNum:=IMAP_Capability(vImap_ID;vCapability)
   ... ` vImap_ID引数を使用するIMAPコマンド
End if
$ErrorNum:=IMAP_Logout(vImap_ID)
```

- このメソッドは以下のメソッドで置き換えられます:

```
$ErrorNum:=IMAP_Login(vHost;vUserName;vUserPassword;vImap_ID)
If($ErrorNum =0)
   C_TEXT(vCapability)
  ` vImap_ID引数を使用するTCPメソッド:
   $ErrorNum:=My_IMAP_Capability(vImap_ID)
```

```
   ... ` vImap_ID引数を使用するTCPコマンド
End if
$ErrorNum:=IMAP_Logout(vImap_ID)
```

- ***My_IMAP_Capability*** 関数のコードは以下の通りです:

```
C_LONGINT($1;$vErrorNum;$0)
C_TEXT($vSentText;$vReceivedText;vCapability)
C_TEXT($2)

$vImap_Id:=$1
$vCmd_Id:="A001" ` このコマンドIDはユニークでなければならない (cf. RFC 2060)
$MyvtRequestCmd:="CAPABILITY"
$vSentText;:=$vCmd_Id+""+$MyvtRequestCmd+"\r\n"
$vReceivedText:=""
$vErrorNum:=TCP_Send($vImap_Id;$vSentText)
If($vErrorNum=0)
   $vErrorNum:=TCP_Receive($vImap_Id;$vReceivedText)
   Case of
      :($vErrorNum#0) `受信エラー
         vCapability:=""
      :(Position($vCmd_Id+" OK ";$vReceivedText)#0)
` コマンド実行に成功
         vCapability:=$vReceivedText
` この例題では受信した文字列を処理しない
      :(Position($vCmd_Id+" BAD ";$vReceivedText)#0)
` コマンド実行に失敗
` (シンタックスエラーまたは未知のコマンド)
         vCapability:=""
         $vErrorNum:=10096
   End case
End if
$0:=$vErrorNum
```

TCP_Close

TCP_Close (tcp_ID) -> 戻り値

引数	型		説明
tcp_ID	倍長整数	➡	開かれたTCPセッションへの参照
		⬅	0 = セッションは閉じられた
戻り値	整数	↩	エラーコード

説明

TCP_Close コマンドは*tcp_ID*で参照されるTCPセッションを閉じます。TCPセッションが閉じられない場合、TCPセッションで利用可能な64セッションのうちのひとつで占有されます。開かれて、閉じられていないセッションが64あると、新しいセッションを開くことはできません。

tcp_ID は、*TCP_Open* または *TCP_Listen* コマンドで確立されたTCPセッション参照です。このコマンドは、セッションが閉じられると、*tcp_ID* 引数に0を返します。

TCP_Listen

TCP_Listen (localAddress ; localPort ; remotePort ; timeout ; tcp_ID) -> 戻り値

引数	型		説明
localAddress	文字	⇒	接続を受け付けるIPアドレス、または""の場合すべてのアドレス
		⇐	使用されたIPアドレス (空の文字列を内容とする変数が渡された場合)
localPort	整数	⇒	ローカルポート番号, 0 = 未使用のポートを検索
		⇐	使用したローカルポート番号 (0を渡した場合)
remotePort	整数	⇒	*** 引数は無視されます ***
timeout	整数	⇒	待ち受け秒数, 0 = 永久に待つ
tcp_ID	倍長整数	⇐	このTCPセッションへの参照
戻り値	整数	⇐	エラーコード

説明

TCP_Listen コマンドは*localAddress*と*localPort*引数で指定したアドレスとポートを使用して通信ソケットを開きます。このコマンドは、接続が確立されるか、*timeout*が経過するまで、4Dにコントロールを返しません。接続が確立されるまでデータベースがロックしたかのように見えますが、このコマンドは他の4Dプロセスに対し有効に振舞います。このコマンドは、既に起動されている4Dプロセスに対し、時間をスライスします。

ほとんどの4Dデベロッパは、独自の4Dプロセスで実行されるメソッドからこのコマンドをコールするでしょう (特に*timeout*をなしにする場合)。

*localAddress*は接続を待ちうけるマシンのIPアドレスです:

- 空の文字列が渡された場合、このコマンドはこのマシン上で利用可能なすべてのアドレスで接続を待ちうけます。
- 空の文字列を含む変数を渡した場合、接続を受け付けたIPアドレスが変数に返されます。

*localPort*には通信に使用するTCPポート番号を渡します。この引数に0を渡すと、コマンドは未使用のポート番号を探し、この引数に返します。

*timeout*には、このコマンドが接続を待ちうける待ち秒数を指定します。この引数に0を渡すと、コマンドは永遠に接続を待ちうけます。0を渡すことには注意が必要です。接続が来ない場合、このコマンドを呼び出した4Dプロセスにはコントロールが返らなくなります。シングルプロセスデータベースの場合、0を渡してはいけません。

tcp_ID には開かれたセッションの参照が返されます。この参照はこのセッションを参照するTCPコマンドで使用されます。

*TCP_Listen*コマンドによるすべてのTCP接続は、*TCP_Close* コマンドで閉じなければなりません。

例題

```
C_LONGINT(vTCPID)
C_INTEGER(vStatus)
$err:=TCP_Listen("";0;0;30;vTCPID)
$err:=TCP_State(vTCPID;vStatus)
If(vStatus=2) `ソケットが開かれ接続を受け付けた
   DoSomething
   $err:=TCP_Close(vTCPID)
End if
```

TCP_Open

TCP_Open (hostName ; remotePort ; tcp_ID ; sessionSettings) -> 戻り値

引数	型		説明
hostName	文字	⇒	ホスト名またはIPアドレス
remotePort	整数	⇒	接続するリモートポート (0 = any)
tcp_ID	倍長整数	⇐	このTCPセッションへの参照
sessionSettings	整数	⇒	TCPセッション設定: 0 = 同期 (省略時のデフォルト); 1 = 非同期; 2 = SSL使用, 同期; 3 = SSL使用, 非同期
戻り値	整数	⇐	エラーコード

説明

TCP_Open コマンドは、ドメインへの外向きのTCP接続を開始します。

*TCP_Open*は、*hostName*で指定されたリモートのTCPへ、指定した*remotePort* (0でない場合) ポート番号で接続を開始します。倍長整数値が*tcp_ID*に返されます。この番号は、このセッションを参照するTCPコマンドで使用されます。*TCP_Open*にはデータを受信しない場合のタイムアウトが30秒に設定されます。デフォルトのタイムアウト値は*IT_SetTimeOut*コマンドで変更できます。

hostName は接続を開くマシンのホスト名またはIPアドレスです。

remotePort は*hostName*で指定したマシンの、接続を行うTCPポートです。

Note: *remotePort*に渡した値が32767より大きい場合、*TCP_Open* (または*TCP_Listen*) 実行後、*remotePort*にマイナスの値が格納されていることがあります。このことは接続に影響ありません。これを避けたい場合、代わりの変数を使用します:

```
$v_ RemotePort:=v_ RemotePort
$err:=TCP_Open(v_ RemoteHostIPAdr;$v_ RemotePort;v_ SessionID)
```

tcp_ID には開かれたセッションへの参照が返されます。この参照は、このセッションを参照するTCPコマンドで使用されます。

sessionSettings はオプションの引数で、TCPセッション設定を指定するための整数値です。この設定は、セッションの間TCPコマンドを実行するたびに適用されます。デフォルト値は 0 (同期, 非SSL) です。

SSL (Secured Socket Layer) はセキュアなTCP接続を行うためのプロトコルです。詳細とインストールに関する要件は4Dのリファレンスを参照してください。

TCP_Open コマンドによるすべてのTCP接続は、*TCP_Close* コマンドで閉じなければなりません。

非同期/同期

非同期 モードは、接続処理の終了を待たず (リモートホストとの接続が確立されるのを待たず) に即座にコントロールを4Dカーネルに返します。非同期モードは、TCPコマンドが4Dの時間を使用することを望まない場合に利用できます。

同期 モードは、接続処理が終了した後 (成功または非成功) に、4Dカーネルにコントロールを返します。

- 0 = 同期モード (デフォルトモード、以前の4D Internet Commandsと同様の動作)
- 1 = 非同期モード
- 2 = SSL使用, 同期。このTCPセッションを参照するすべてのTCPコマンドは、SSLプロトコルを使用して、同期モード

で動作します。
- 3 = SSL使用, 非同期。このTCPセッションを参照するすべてのTCPコマンドは、SSLプロトコルを使用して、非同期モードで動作します。

Note: 2または3を渡した場合、SSL接続を開くことができないとエラー10089が返されます (SLIライブラリが4D Extensionsフォルダに見つからない)。

例題

HTTPSを使用してWebサイトに接続します。SLIが正しくインストールされていることを確認し、ポート番号443を使用して接続を行います:

```
$vError:=TCP_Open(hostName;443;tcp_ID;2)
...
$vError:=TCP_Close(tcp_ID)  `セッションを閉じることを忘れないように
```

TCP_Receive

TCP_Receive (tcp_ID ; text) -> 戻り値

引数	型		説明
tcp_ID	倍長整数	➡	TCPセッション参照
text	テキスト	⬅	受信したテキスト
戻り値	整数	↩	エラーコード

説明

確立されたTCPセッション参照を指定し、*TCP_Receive* コマンドはtextにパケットデータを受け取ります。

tcp_ID は、*TCP_Open* または *TCP_Listen* コマンドで確立されたTCPセッション参照です。

text には受信したテキストが格納されます。TCPパケットのデータを受信するとき、一回の*TCP_Receive*の呼び出しですべてのデータを受信できるとは限りません。*TCP_Receive* コマンドは通常、接続ステータスや特定の値をチェックしながら、*Repeat*ループの中で使用されます。

例題

```
C_LONGINT($tcp_id)
C_TEXT($webpage;$buffer)
C_INTEGER(vState;$error)
$webpage:=""
vState:=0
Repeat
   $error:=TCP_Receive($tcp_id;$buffer)
   $error:=TCP_State($tcp_id;vState)
   $webpage:=$webpage+$buffer
Until((vState=0)|($error#0))  ` ホストが接続を閉じるかエラーが発生するまで
```

TCP_ReceiveBLOB

TCP_ReceiveBLOB (tcp_ID ; blobToReceive) -> 戻り値

引数	型		説明
tcp_ID	倍長整数	➡	TCPセッション参照
blobToReceive	BLOB	⬅	受信したデータを格納するBLOB
戻り値	整数	↩	エラーコード

説明

確立されたTCPセッション参照を指定し、*TCP_ReceiveBLOB* コマンドは*blobToReceive*にパケットデータを受け取ります。

tcp_ID は、*TCP_Open* または *TCP_Listen* コマンドで確立されたTCPセッション参照です。

このコマンドは、32kの制限があるテキストの代わりにデータをBlobに受信すること以外、*TCP_Receive*コマンドと同じ動作を行います。このコマンドを使用するとバイナリオブジェクトを受信できるようになります。

blobToReceive には受信したBLOBが格納されます。TCPパケットのデータを受信するとき、一回の*TCP_ReceiveBLOB*の呼び出しですべてのデータを受信できるとは限りません。*TCP_ReceiveBLOB* コマンドは通常、接続ステータスや特定の値をチェックしながら、*Repeat*ループの中で使用されます。

例題

この例題では *TCP_ReceiveBLOB*を使用する典型的な構造を紹介します:

```
C_BLOB($Blob_Received;$Blob_All)
C_LONGINT($srcpos;$dstpos)
Repeat
   $Err:=TCP_ReceiveBLOB($TCP_ID;$Blob_Received )
   $Err:=TCP_State($TCP_ID;$State)
   $srcpos:=0
   $dstpos:=BLOB size($Blob_All)
  `受信したBLOBを連結する
   COPY BLOB($Blob_Received;$Blob_All;$srcpos;$dstpos;BLOB size($Blob_Received))
Until(($State=0)|($Err#0))
```

TCP_Send

TCP_Send (tcp_ID ; sendText) -> 戻り値

引数	型		説明
tcp_ID	倍長整数	➡	TCPセッション参照
sendText	テキスト	➡	送信するテキスト
戻り値	整数	⮌	エラーコード

説明

TCP_Send コマンドは、*tcp_ID*で指定したTCPセッションにデータを送信します。

tcp_ID は、*TCP_Open* または *TCP_Listen* コマンドで確立されたTCPセッション参照です。

sendText はTCPセッションに送信するテキストです。

TCP_SendBLOB

TCP_SendBLOB (tcp_ID ; blobToSend) -> 戻り値

引数	型		説明
tcp_ID	倍長整数	➡	TCPセッション参照
blobToSend	BLOB	➡	送信するBlob
戻り値	整数	⮌	エラーコード

説明

TCP_SendBLOB コマンドは、*tcp_ID*で参照されるTCPセッションにデータを送信します。このコマンドは、32kの制限があるテキストの代わりにBlobを送信すること以外、*TCP_Send*コマンドと同じ動作を行います。このコマンドを使用するとバイナリオブジェクトを送信できるようになります。

tcp_ID は、*TCP_Open* または *TCP_Listen* コマンドで確立されたTCPセッション参照です。

blobToSend は、*tcp_ID*で参照されるTCPセッションに送信するBLOBです。

プラットフォーム独立性に関する注意: MacintoshとPCプラットフォーム間でBLOBを送受信する場合、必要に応じてバイトスワッピングを管理するのは開発者の仕事です。

例題

この例題ではTCPセッションにBLOBを送信します:

```
C_BLOB($Blob_Send)
C_TEXT(v_Txt_Send)
TEXT TO BLOB(v_Txt_Send;$Blob_Send;Text without length;*)
$err:=TCP_SendBLOB(v_tcp_ID;$Blob_Send)
```

TCP_State

TCP_State (tcp_ID ; statusCode) -> 戻り値

引数	型		説明
tcp_ID	倍長整数	➡	TCP接続参照
statusCode	整数	⬅	TCPステータスコード
戻り値	整数	➲	エラーコード

説明

TCP_State コマンドは特定のTCP接続のステータスに対応する整数値を返します。

tcp_ID は、*TCP_Open* または *TCP_Listen* コマンドで確立されたTCPセッション参照です。

statusCode には以下のステータスコードのいずれかが返されます。

0	接続は閉じている
2	接続待ち受け中
8	接続確立

例題

この例題では、有効なTCP接続が確立されていて、*$tcp_id*に接続参照が含まれているとします。この例題では、コマンドはサーバにWebページをリクエストし、結果を受け取るためループに入ります。Webサーバはレスポンス送信後に自動で接続を閉じるため、この例題は接続が閉じられるかエラーが発生するまで受信を続けます。

```
C_LONGINT($tcp_id)
C_INTEGER(vState;$err)
C_TEXT($command;$buffer;$response)
If(TCP_Send($tcp_id;$command)=0)
   vState:=0
   Repeat
      $err:=TCP_Receive($tcp_id;$buffer)
      $err:=TCP_State($tcp_id;vState)
      $response:=$response+$buffer
   Until((vState=0)|($err#0))
End if
```

IC UDP

- UDPコマンド - 概要
- UDP_Delete
- UDP_New
- UDP_ReceiveBLOBFrom
- UDP_ReceiveFrom
- UDP_SendBLOBTo
- UDP_SendTo

UDPコマンド - 概要

UDP (User Datagram Protocol) は実装が簡単なデータ送信プロトコルです。TCPに比べ高速でシンプルです (TCPの場合最低20バイトがヘッダに必要なのに対し、UDPは8バイトです) が、信頼性は提供されません。このプロトコルはデータが素早く送信先に到達することが重要なアプリケーションで使用できます。このプロトコルはデータ転送 検証やエラーチェック、送信されなかったデータの復元ができません。

例題

この例題は、UDPコマンドを使用してローカルネットワークで実行されている4D Serverのリストを取得する方法を示しています:

```
ARRAY TEXT(asHost;0)
ARRAY TEXT(asMachineName;0)
ARRAY TEXT(asService;0)
ARRAY TEXT(asDBName;0)
C_BLOB($Blob)

$Addr:="255.255.255.255"
$Port:=19813
$Offset:=32
SET BLOB SIZE($Blob;96;0)
TEXT TO BLOB("4D Server II";$Blob;Mac Text without length;$Offset)

$Err:=UDP_New(0;$udpID)
$Err:=UDP_SendBLOBTo($udpID;$Addr;$Port;$Blob)
$Secs:=5
$Timeout:=Milliseconds+($Secs*1000)
Repeat
   DELAY PROCESS(Current process;6) `... in ticks
   SET BLOB SIZE($Blob;0;0)
   $PeerAddr:=$Addr
   $Err:=UDP_ReceiveBLOBFrom($udpID;$PeerAddr;$Port;$Blob)

   If(BLOB size($Blob)>0)
      $Offset:=0
      $Host:=BLOB to text($Blob;Mac C string;$Offset;32)
      $Offset:=32
      $Service:=BLOB to text($Blob;Mac C string;$Offset;32)
      $Offset:=64
      $DBName:=BLOB to text($Blob;Mac C string;$Offset;32)
      $Pos:=Find in array(asMachineName;$Host)
      If($Pos=-1)
         APPEND TO ARRAY(asHost;$PeerAddr)
         APPEND TO ARRAY(asMachineName;$Host)
         APPEND TO ARRAY(asService;$Service)
         APPEND TO ARRAY(asDBName;$DBName)
```

```
      End if
   End if
Until((Milliseconds>$Timeout)|($Err#0))
$Err:=UDP_Delete($udpID)
```

UDP_Delete

UDP_Delete (udp_ID) -> 戻り値

引数	型		説明
udp_ID	倍長整数	➡	UDPソケット参照
戻り値	整数	⮌	エラーコード

説明

UDP_Delete コマンドは*udp_ID*を指定して、*UDP_New* を使用して開かれたUDPソケットを閉じるために使用します。

UDP_New

UDP_New (localPort ; udp_ID) -> 戻り値

引数	型		説明
localPort	整数	⇒	UDPソケットで使用するローカルポート (0 = 未使用のポートを検索)
udp_ID	倍長整数	⇐	UDPソケット参照
戻り値	整数	↩	エラーコード

説明

UDP_New コマンドはUDPソケットを作成するために使用します。*localPort*引数には使用するローカルポート番号を渡します。0を渡すと、コマンドは未使用のポートを検索します。UDPソケット参照が*udp_ID* 引数に返されます。このソケットが必要なくなった時は、メモリーを解放するために*UDP_Delete*を使用してソケットを閉じます。

UDP_ReceiveBLOBFrom

UDP_ReceiveBLOBFrom (udp_ID ; hostName ; remotePort ; BLOB) -> 戻り値

引数	型		説明
udp_ID	倍長整数	→	UDPソケット参照
hostName	文字	→	サーバ名またはIPアドレス
remotePort	整数	→	接続するリモートポート (0=any)
BLOB	BLOB	←	受信したBLOB
戻り値	整数	↩	エラーコード

説明

UDP_ReceiveBLOBFrom コマンドは、*udp_ID*ソケットを通じて送信されたBLOBを受信するために使用します。

*hostName*はBLOBを受信するサーバのIPアドレスまたは名前です。

*remotePort*は接続するポート番号です。0を渡すと利用可能なポートが使用されます。

受信したBLOBは*blob*引数に返されます。

UDP_ReceiveFrom

UDP_ReceiveFrom (udp_ID ; hostName ; remotePort ; text) -> 戻り値

引数	型		説明
udp_ID	倍長整数	⇒	UDPソケット参照
hostName	文字	⇒	サーバ名またはIPアドレス
remotePort	整数	⇒	接続先のリモートポート (0=any)
text	テキスト	⇐	受信したテキスト
戻り値	整数	↩	エラーコード

説明

UDP_ReceiveFrom コマンドは、*udp_ID*ソケットを通してテキストを受信するために使用します。

hostName テキストを受信するサーバの名前またはIPアドレスです。

remotePort は接続するポート番号です。0を渡すと利用可能なポートが使用されます。

受信したテキストは*text* 引数に返されます。

UDP_SendBLOBTo

UDP_SendBLOBTo (udp_ID ; hostName ; remotePort ; sendBlob) -> 戻り値

引数	型		説明
udp_ID	倍長整数	→	UDPソケット参照
hostName	文字	→	サーバの名前またはIPアドレス
remotePort	整数	→	接続するリモートポート (0=any)
sendBlob	BLOB	→	送信するBLOB
戻り値	整数	↩	エラーコード

説明

UDP_SendBLOBTo コマンドは*udp_ID*ソケットを使用してBLOBを送信するために使用します。

hostName はBLOB送信先のサーバの名前またhIPアドレスです。

remotePort は接続するポート番号です。0を渡すと利用可能なポートが使用されます。

sendBlob は送信するBLOBです。

UDP_SendTo

UDP_SendTo (udp_ID ; hostName ; remotePort ; sendText) -> 戻り値

引数	型		説明
udp_ID	倍長整数	→	UDPソケット参照
hostName	文字	→	サーバ名またはIPアドレス
remotePort	整数	→	接続先のリモートポート (0=any)
sendText	テキスト	→	送信するテキスト
戻り値	整数	↩	エラーコード

説明

UDP_SendTo コマンドは、*udp_ID* ソケットを使用してテキストデータを送信するために使用します。

hostName は、テキストを送信する先のサーバの名前またはIPアドレスです。

remotePort は接続先のポート番号です。0を渡すと利用可能なポートが使用されます。

sendText は送信するテキストです。

IC インターネット

- 特別なインターネットコマンド - 概要
- NET_AddrToName
- NET_Finger
- NET_NameToAddr
- NET_Ping
- NET_Resolve
- NET_Time

特別なインターネットコマンド - 概要

この節に含まれるコマンドは、インターネットで一般的なタスクを行うために使用します。 この節に含まれるコマンドにはマシンに対する'Ping'や'Finger'、時刻サーバからの時間の取得、ドメイン名やIPアドレスの解決、ドメイン名 やIPアドレスを倍長整数値に変換したりまたはその逆を行うものが含まれます。これらのコマンドはしばしば他の4D Internet Commandsとともに使用されます。

NET_AddrToName

NET_AddrToName (ip_Longint ; hostName ; ip_Address) -> 戻り値

引数	型		説明
ip_Longint	倍長整数	⇒	アドレスへの倍長整数参照
hostName	文字	⇐	ホスト名
ip_Address	文字	⇐	IPアドレス
戻り値	整数	⇐	エラーコード

説明

NET_AddrToName コマンドはホスト名の倍長整数参照をホスト名とIPアドレスに変換します。

ip_Longint はIPアドレスの倍長整数参照です。

hostName にはホスト名文字列が返されます。ホスト名の解決ができない場合、*ip_Address*には空の文字列が返され、エラーは発生しません。

ip_Address にはIPアドレスが返されます。

NET_Finger

NET_Finger (hostName ; searchText ; results) -> 戻り値

引数	型		説明
hostName	文字	⇒	ホスト名またはIPアドレス
searchText	文字	⇒	検索テキスト
results	テキスト	⇐	Fingerの結果
戻り値	整数	⇒	エラーコード

説明

検索するマシンのIPアドレスと、マシンのユーザアカウント名を渡すと、*NET_Finger* コマンドは*results*にFingerの結果を返します。Unixのfinger コマンドは、ユーザが最後にログインした時間や、 ".plan" や ".project" ファイルに記述された追加のユーザ情報を取得するために使用します。

Finger検索で使用するために、二つの異なるルートを指定できます。Finger検索は直接ユーザマシン上で試みられます。たとえば"4d.com"における"johnt"の情報を取得するために、以下の検索を行うことができます:

```
$error:=NET_Finger("www.4d.com";"johnt";$fingertext)
```

同じFinger検索を間接的に行うこともできます。間接検索は、検索を実行するFingerコマンドをサポートするリモートサーバに依頼します。例えば、以下の検索はドメイン名"4d.com"で指定されるマシンに、"johnt@4d.com"のリモートFingerの実行を依頼します。.

```
$error:=NET_Finger("www.4d.com";"johnt@4d.com";$fingertext)
```

それぞれの場合で返される主な情報は同じであるべきであるため、返されるテキストには少々の違いしかありません。Fingerコマンド実行時に、異なるマシンでは異なるオプションが設定されている場合があり、結果は多少異なります。また直接または間接の結果には、フォーマットの相違があり、関節の場合にはしばしばラインフィードが追加されます。

hostName は、*searchText*で指定されるユーザががアカウントを持つマシンのホスト名またはIPアドレスです。

searchText は、与えられたFingerサーバ上の検索テキストまたはマシン名やIPアドレスです。*searchText*がユーザ名の場合、コマンドはサーバ上でユーザ名のディレクトリを検索します。*searchText*がマシン名またはIPアドレスの場合、コマンドはFingerリクエストを、Fingerサーバを通じて、*hostName*で指定したマシンにFingerリクエストを送信します。

results には検索結果テキストが返されます。

NET_NameToAddr

NET_NameToAddr (hostName ; ip_Longint) -> 戻り値

引数	型		説明
hostName	文字	⇒	ホスト名またはIPアドレス
ip_Longint	倍長整数	⇐	アドレスへの倍長整数参照
戻り値	整数	⇒	エラーコード

説明

NET_NameToAddr コマンドはホスト名またはIPアドレスを受け取り、アドレスへのユニークな倍長整数参照を渡します。

hostName はホスト名またはIPアドレスです。

ip_Longint は、*hostName* 引数で指定されたIPアドレスの倍長整数値です。すべてのIPアドレス文字列は倍長整数値に変換できます。

*ip_Longint*値に特記すべき利用方法はありませんが、時にデータを格納するのにコンパクトな倍長整数値を利用したい場合に利用できます。

NET_Ping

NET_Ping (hostName ; text ; alive ; timeout) -> 戻り値

引数	型		説明
hostName	文字	→	ホスト名またはIPアドレス
text	テキスト	→	Pingで送信するテキスト
alive	整数	←	1 = アクティブ, 0 = タイムアウト/レスポンスなし
timeout	整数	→	待ち秒数, 0 = IT_SetTimeOut値を使用
戻り値	整数	↩	エラーコード

説明

NET_Ping コマンドは、IPアドレスで指定するマシンが現在アクティブかどうかを知るためのメカニズムを提供します。Pingを打たれたマシンが現在TCP/IPプロトコルを実行していて、二つのマシン間のネットワークが機能していれば、'レスポンスあり'ステータスが返されます。

NET_Ping はホスト名またはIPアドレスで指定されたマシンにPingを送信します。ネットワークでアクセス可能なマシンにIPアドレスでPingを送信できます。これにはエンドユーザのマシンが含まれます (ファイアウォールによる保護のためPingが妨げられることがあります) 。

hostName はPingの送信先ホスト名またはIPアドレスです。

text はPingで送信するテキストです。text 引数は、Pingコマンドが実行される際に送信されるTCPパケットのサイズにのみ影響を与えます。

alive にはPingされたマシンのステータスが返されます。1はマシンがアクティブであることを示します。0はマシンがアクティブでないか、レスポンスを受け取る前にタイムアウトしたことを示します。

timeout は、このコマンドがPing完了までに待ち受ける秒数を指定します。これはオプションの引数で省略可能です。デフォルト値は0で、この場合、*IT_SetTimeOut* コマンドで設定された値が使用されあます。

Note: この引数はWindows 95/98 および Millenniumでは考慮されません。

NET_Resolve

NET_Resolve (hostName ; ipOrHost) -> 戻り値

引数	型		説明
hostName	文字	⇒	ホスト名またはIPアドレス
ipOrHost	文字	⇐	対応するIPアドレスまたはホスト名
戻り値	整数	➲	エラーコード

説明

一番目の引数にホスト名を渡すと、*NET_Resolve* コマンドは二番目の引数にIPアドレスを返します。一番目の引数にIPアドレスを渡すと、二番目の引数にはマシンの登録された名前が返されます。

hostName はIPアドレスまたはホスト名文字列です。

ipOrHost: 一番目の引数にホスト名が渡された場合、この引数にはIPアドレスが返されます。IPアドレスが一番目の引数に渡されると、ホスト名が返されます。

例題

以下の例題ではまずホスト名"www.netcom.com"を*NET_Resolve* コマンドに渡し、IPアドレスを取得しています。その後、もう一度コマンドを呼び出して、IPアドレスから登録されたホスト名を取得しています。

```
C_BOOLEAN($ERR)
C_TEXT($Resolved)
$ERR:=ERRCHECK("NET_Resolve";NET_Resolve("www.netcom.com";$Resolved))
  `$Resolvedに'192.100.81.100'が返される
$ERR:=ERRCHECK("NET_Resolve";NET_Resolve($Resolved;$Resolved))
  `$Resolvedに'www.netcom.com'が返される
```

NET_Time

NET_Time (hostName ; date ; time ; offset) -> 戻り値

引数	型		説明
hostName	文字	➡	ホスト名またはIPアドレス
date	日付	⬅	日付
time	倍長整数	⬅	0時からの秒数で表現される時間
offset	整数	➡	オフセットする時間
戻り値	整数	↩	エラーコード

説明

インターネット時刻サーバのホスト名またはIPアドレスを渡すと、*NET_Time* コマンドは、現在の日付と時間をサーバから受け取り、オフセットを適用してユーザのローカル時間に変換します。

Note:

- このコマンドはコンピュータの内部時計には影響しません。
- このコマンドはポート番号37のTimeサーバにアクセスします。

hostName はインターネット時刻サーバのホスト名またはIPアドレスです。

Date には、*offset*適用後の結果の4D日付が返されます。

Time には、*offset*適用後の結果の時間が返されます。値は*Date* の0時からの経過秒数で表現されます。例題でこの秒数を4Dの時間に変換する方法を示しています。

offset は、基本の時間値に追加あるいは減じる時間数を渡します。インターネット時刻サーバは値をユニバーサル時刻で返します。時刻サーバがローカルの地域にあるとしても、ユニバーサル時刻とローカル時刻の時差を指定する必要があります。

例題

以下の例題は"apple.com"の時刻サーバからユニバーサル時刻を取得します。このコマンドはそののち、オフセットで指定した7時間を減じ、結果の日付と時刻を返します。 (時間は倍長整数値で表現され、4DのTime stringコマンドで変換できます。)

```
C_DATE(vNetDate)
C_LONGINT(vNetTime)
C_TIME(vTime)
C_INTEGER(vOffset)
If(ERRCHECK("NET_Time";NET_Time("www.apple.com";vNetDate;vNetTime;-7)))
   vTime:=Time(Time string(vNetTime))  `倍長整数表現の時刻を4D時間に変換
End if
```

IC ダウンロードしたメール

- メールのダウンロード - 概要
- MSG_Charset
- MSG_Delete
- MSG_Extract
- MSG_FindHeader
- MSG_GetBody
- MSG_GetHeaders
- MSG_GetMessage
- MSG_GetPrefs
- MSG_HasAttach
- MSG_MessageSize
- MSG_SetPrefs

メールのダウンロード - 概要

"MSG_"から始まるコマンドは、*POP3_Download* や *IMAP_Download* コマンドでローカルファイルとして保存したメールメッセージを操作するために使用します。これらのコマンドは完全にMIME準拠であり、4D Internet Commandsを使用して添付ファイルを取り出したりデコードしたりできます。MIME標準に関する詳細は、RFC#1521, RFC#1522, RFC#2045を参照してください。

メッセージをローカルファイルにダウンロードした後、このセクションのコマンドを使用してドキュメントに対し様々な処理を行うことができます。これらのコマンドを使用してメッセージパートを取得したり、ヘッダのみを取り出したり、添付ファイルを取り出したり、既存のドキュメントを削除したりできます。

MSG_Charset

MSG_Charset (decodeHeaders ; bodyCharset) -> 戻り値

引数	型		説明
decodeHeaders	整数	➡	-1 = 現在の設定を使用, 0 = 管理しない, 1 = ISO-8859-1またはISO-2022-JPの場合Mac OS文字セットを使用して変換、拡張文字をデコード
bodyCharset	整数	➡	-1 = 現在の設定を使用, 0 = 管理しない, 1 = ISO-8859-1 または ISO-2022-JPの場合Mac OS文字セットを使用して変換、拡張文字をデコード
戻り値	整数	↩	エラーコード

説明

MSG_Charset コマンドを使用して、MSGコマンドによる拡張文字を含むメッセージ処理を自動化できます。このコマンドを使用しないか0を指定した場合、4D Internet Commandsはバージョン6.5.xと同様に動作します。

MSG_Charset はまず、拡張文字ヘッダーのデコードに関する設定を行い、次にメッセージボディとヘッダーの文字セット変換を行うかどうかの設定を行います。

このコマンドは特に"Subject"やメールアドレスなどのヘッダーに拡張文字が含まれるメッセージのサポートに有用です (例えば“*=?ISO-8859-1?Q?Test=E9?= <test@n.net >*”のようなメールアドレス)。

decodeHeaders 引数は*MSG_FindHeader*実行中にヘッダーのデコードと変換をどのように行うかを指定します。デフォルトで0に設定されています。

- -1: 現在の設定を使用
- 0: 管理しない
- 1: 必要に応じてヘッダーをデコードする。デコードが行われ、かつ文字セットがISO-8859-1またはISO-2022-JPである場合、ヘッダーは4Dテキストに自動で変換されます。

bodyCharset 引数は*MSG_GetBody* 実行中にボディの変換をどのように行うかを指定します。デフォルトで0に設定されています。

- -1: 現在の設定を使用
- 0: 管理しない
- 1: “Body-Content-Type”文字セットがISO-8859-1またはISO-2022-JPに設定されていれば、メッセージボディは4Dテキストに自動で変換されます。

互換性メモ (バージョン 6.8.1): *MSG_Charset* コマンドが使用されず*POP3_Charset* コマンドが使用されていた場合、*MSG_FindHeader*と*MSG_GetBody*コマンドは*POP3_Charset*の設定を考慮します。*MSG_Charset*が使用されていた場合、*POP3_Charset*の設定は無視されます。

例題 1

バージョン6.5.x の 4D Internet Commands:

```
$Err:=MSG_FindHeader($msgfile;"From";$from)
$from:=ISO to Mac($from)
$Err:=MSG_FindHeader($msgfile;"To";$to)
$to:=ISO to Mac($to)
$Err:=MSG_FindHeader($msgfile;"Cc";$cc)
$cc:=ISO to Mac($cc)
$Err:=MSG_FindHeader($msgfile;"Subject";$subject)
$subject:=ISO to Mac($subject)
```

```
$Err:=MSG_MessageSize($msgfile;$HdrSize;$BdySize;$msgSize)
$Err:=MSG_GetBody($msgfile;0;$BdySize;$BodyContent)
$BodyContent:=ISO to Mac($BodyContent)
```

例題 2

バージョン 6.8.1 以降の 4D Internet Commands:

```
$Err:=MSG_Charset(1;1)
$Err:=MSG_FindHeader($msgfile;"From";$from)
$Err:=MSG_FindHeader($msgfile;"To";$to)
$Err:=MSG_FindHeader($msgfile;"Cc";$cc)
$Err:=MSG_FindHeader($msgfile;"Subject";$subject)
$Err:=MSG_MessageSize($msgfile;$HdrSize;$BdySize;$msgSize)
$Err:=MSG_GetBody($msgfile;0;$BdySize;$BodyContent).
```

MSG_Delete

MSG_Delete (fileName ; folder) -> 戻り値

引数	型		説明
fileName	テキスト	⇒	ファイル名
folder	整数	⇒	0 = メッセージフォルダ, 1 = 添付フォルダ
戻り値	整数	⇐	エラーコード

説明

MSG_Delete コマンドはローカルファイルを削除します。

fileName は削除するファイルのファイル名またはパス名です。ファイル名のみが指定された場合、*folder*引数が以下の要領で考慮されます:

- *folder* = 0: ファイルは*POP3_SetPrefs*または*MSG_SetPrefs*で指定されたメッセージフォルダに存在する。
- *folder* = 1: ファイルは*POP3_SetPrefs*または*MSG_SetPrefs*で指定された添付フォルダに存在する。

両ケースとも、*POP3_SetPrefs*または*MSG_SetPrefs*でフォルダが指定されていない場合、ストラクチャと同階層 (4D シングルユーザ) または4Dクライアントフォルダ (4D Server) でファイルを探します。

互換性メモ (バージョン6.8.1): *MSG_SetPrefs* コマンドを使用しない場合、*POP3_SetPrefs*の*msgFolder*と*attachFolder*引数が考慮されます。*MSG_SetPrefs*が使用されていれば*POP3_SetPrefs*の*msgFolder*と*attachFolder*引数は無視されます。

警告： このコマンドは指定されたファイルを削除します。利用の際は十分注意してください。

MSG_Extract

MSG_Extract (fileName ; decode ; attachmentPath ; enclosureList) -> 戻り値

引数	型		説明
fileName	テキスト	⇒	ファイル名
decode	整数	⇒	0 = デコードしない, 1 = 可能であればデコードする
attachmentPath	テキスト	⇒	フォルダパス
enclosureList	文字配列	⇐	同封ファイル名 (パスなし)
戻り値	整数	↩	エラーコード

説明

MSG_Extract コマンドはすべての添付を展開し、添付フォルダに置きます。

fileName は、ヘッダ情報を取得するファイルの名前またはファイルのフルパスです。ファイル名のみを指定した場合、*POP3_SetPrefs* または *MSG_SetPrefs* で指定したパスでファイルを検索します (互換性メモ参照)。フォルダが指定されていない場合、ストラクチャと同階層 (4D シングルユーザ) または4Dクライアントフォルダ (4D Server) でファイルを探します。

*decode*は添付をデコードするかしないか指定する整数値です。0を指定すると、添付をデコードしません。1 を指定すると、以下のいずれかの方法でエンコードされている場合、デコードを行います: Binhex, AppleSingle, AppleDouble, Base64.

attachmentPath は添付を格納するフォルダパスです。フォルダパスが指定されていない場合、ファイルは*POP3_SetPrefs* または *MSG_SetPrefs* (互換性メモ参照) で指定されたフォルダに保存されます。フォルダパスが定義されていない場合、添付はデータベースストラクチャと同階層に格納されます。

互換性メモ (バージョン6.8.1): *MSG_SetPrefs* コマンドを使用しない場合、*POP3_SetPrefs*の*msgFolder*と*attachmentPath*引数が使用されます。*MSG_SetPrefs*が使用されていれば*POP3_SetPrefs*の*msgFolder*と*attachmentPath*引数は無視されます。

enclosureList はテキスト/文字列配列で、添付のファイル名が返されます。パス名を含まない、ファイル名のみが返されます。

MSG_FindHeader

MSG_FindHeader (fileName ; headerLabel ; headerValue) -> 戻り値

引数	型		説明
fileName	テキスト	⇒	ファイル名
headerLabel	文字	⇒	ヘッダラベル ("From:", "To:", "Subject:", その他)
headerValue	テキスト	⇐	値
戻り値	整数	↩	エラーコード

説明

POP3_Download や *IMAP_Download* コマンドで取得したメッセージドキュメントの*fileName*を指定して、*MSG_FindHeader* コマンドは*headerLabel*で指定したヘッダをヘッダセクションで検索し、*headerValue*にその値を返します。

fileName は、ヘッダ情報を取得するファイルの名前またはファイルのフルパスです。ファイル名のみを指定した場合、*POP3_SetPrefs* または *MSG_SetPrefs* で指定したパスでファイルを検索します (互換性メモ参照)。フォルダが指定されていない場合、ストラクチャと同階層 (4D シングルユーザ) または4Dクライアントフォルダ (4D Server) でファイルを探します。

互換性メモ (バージョン6.8.1): *MSG_SetPrefs* コマンドを使用しない場合、*POP3_SetPrefs*の*msgFolder* 引数が使用されます。*MSG_SetPrefs*が使用されていれば*POP3_SetPrefs*の*msgFolder* 引数は無視されます。

headerLabel はヘッダラベル文字列です。*headerLabel* には"From:", "To:", "X-MyHeader"などの定義済み、ユーザ定義、および拡張ヘッダを指定可能です。

headerValue は指定したヘッダの値を受け取るテキスト変数です。*headerValue* 引数は拡張文字を受け取ることがあります。*POP3_Charset* や *MSG_Charset* コマンドを使用して処理を自動化することができます。

互換性メモ (バージョン6.8.1): *MSG_Charset* コマンドを使用しない場合、*POP3_Charset*の*bodyCharset* 引数が使用されます。*MSG_Charset*が使用されていれば*POP3_Charset*の*bodyCharset* 引数は無視されます。

MSG_GetBody

MSG_GetBody (fileName ; offset ; length ; bodyText) -> 戻り値

引数	型		説明
fileName	テキスト	⇒	ファイル名
offset	倍長整数	⇒	ボディ中の開始位置オフセット (0 = ボディの先頭)
length	倍長整数	⇒	文字数
bodyText	テキスト	⇐	ボディテキスト
戻り値	倍長整数	⇐	エラーコード

説明

MSG_GetBody コマンドはメッセージテキストを返します。添付ファイルやMIMEヘッダは取り除かれます。

fileName は、ヘッダ情報を取得するファイルの名前またはファイルのフルパスです。ファイル名のみを指定した場合、*POP3_SetPrefs* または *MSG_SetPrefs* で指定したパスでファイルを検索します (互換性メモ参照)。フォルダが指定されていない場合、ストラクチャと同階層 (4D シングルユーザ) または4Dクライアントフォルダ (4D Server) でファイルを探します。

offset は読み込みを開始するソースボディ中の文字位置です。

length は読み込む文字数です。

bodyText にはボディテキストが返されます。*bodyText* には拡張文字が含まれることがあります。*POP3_Charset* や *MSG_Charset* コマンドを使用してこの管理を自動化できます (互換性メモ参照)。この引数は*POP3_SetPrefs* や *MSG_SetPrefs*で設定された*stripLineFeed* 引数を考慮します (互換性メモ参照)。

互換性メモ (バージョン6.8.1):

- *MSG_SetPrefs* コマンドを使用しない場合、*POP3_SetPrefs*の*msgFolder*と*stripLineFeed* 引数が考慮されます。*MSG_SetPrefs*が使用されていれば*POP3_SetPrefs*の*msgFolder*と*stripLineFeed* 引数は無視されます。
- *MSG_Charset* コマンドを使用しない場合、*POP3_Charset*の*bodyCharset* 引数が使用されます。*MSG_Charset*が使用されていれば*POP3_Charset*の*bodyCharset* 引数は無視されます。

MSG_GetHeaders

MSG_GetHeaders (fileName ; offset ; length ; headerText) -> 戻り値

引数	型		説明
fileName	テキスト	⇒	ファイル名
offset	倍長整数	⇒	ヘッダ中の開始位置オフセット (0 = ヘッダの先頭)
length	倍長整数	⇒	文字数
headerText	テキスト	⇐	ヘッダテキスト
戻り値	整数	↩	エラーコード

説明

MSG_GetHeaders コマンドはメッセージのヘッダセクション全体を返します。POP3メッセージおけるヘッダはメッセージの先頭から、最初に見つかった二つの連続するCR/LFまでと定義されています。

fileName は、ヘッダ情報を取得するファイルの名前またはファイルのフルパスです。ファイル名のみを指定した場合、*POP3_SetPrefs* または *MSG_SetPrefs* で指定したパスでファイルを検索します (互換性メモ参照)。フォルダが指定されていない場合、ストラクチャと同階層 (4D シングルユーザ) または4Dクライアントフォルダ (4D Server) でファイルを探します。

offset は読み込みを開始するソースヘッダ中の文字位置です。

length は読み込む文字数です。ヘッダ長は*MSG_MessageSize*で取得できます。

headerText にはヘッダテキストが返されます。この引数は*POP3_SetPrefs* や*MSG_SetPrefs*で設定された*stripLineFeed* 引数を考慮します。

互換性メモ (バージョン6.8.1): *MSG_SetPrefs* コマンドを使用しない場合、*POP3_SetPrefs*の*msgFolder* 引数が使用されます。*MSG_SetPrefs*が使用されていれば*POP3_SetPrefs*の*msgFolder* 引数は無視されます。

MSG_GetMessage

MSG_GetMessage (fileName ; offset ; length ; rawText) -> 戻り値

引数	型		説明
fileName	テキスト	⇒	ファイル名
offset	倍長整数	⇒	メッセージファイル中の開始位置オフセット (0 = ファイルの先頭)
length	倍長整数	⇒	文字数
rawText	テキスト	⇐	未加工のテキスト (環境設定を無視)
戻り値	整数	⇐	エラーコード

説明

MSG_GetMessage コマンドは添付も含め、未加工のメッセージテキストを返します。MIMEヘッダは取り除かれません。

fileName は、ヘッダ情報を取得するファイルの名前またはファイルのフルパスです。ファイル名のみを指定した場合、*POP3_SetPrefs* または *MSG_SetPrefs* で指定したパスでファイルを検索します (互換性メモ参照)。フォルダが指定されていない場合、ストラクチャと同階層 (4D シングルユーザ) または4Dクライアントフォルダ (4D Server) でファイルを探します。

互換性メモ (バージョン6.8.1): *MSG_SetPrefs* コマンドを使用しない場合、*POP3_SetPrefs*の*msgFolder* 引数が使用されます。*MSG_SetPrefs*が使用されていれば*POP3_SetPrefs*の*msgFolder* 引数は無視されます。

offset は読み込みを開始するソースメッセージ中の文字位置です。

length は読み込む文字数です。

rawText はメッセージ全体のテキストを受け取ります。ラインフィードを取り除く環境設定は無視され、メッセージボディに組み込まれた添付も取り除かれません。

MSG_GetPrefs

MSG_GetPrefs (stripLineFeed ; msgFolder ; attachFolder) -> 戻り値

引数	型		説明
stripLineFeed	整数	⇦	0 = LFを保持する, 1 = LFを取り除く
msgFolder	テキスト	⇦	メッセージフォルダパス ("" = 変更しない)
attachFolder	テキスト	⇦	添付フォルダパス ("" = 変更しない)
戻り値	整数	⇨	エラーコード

説明

MSG_GetPrefs コマンドは現在のMSGコマンドの環境設定値を返します。

環境設定値は引数にリストした変数に返されます。

stripLineFeed にはラインフィードの処理に関する設定が返されます。

msgFolder には、取得したメッセージが格納されているデフォルトフォルダを示すパス名が返されます。

attachFolder には、展開した添付のデフォルト保存先フォルダのパスが返されます。

MSG_HasAttach

MSG_HasAttach (fileName ; attachCount) -> 戻り値

引数	型		説明
fileName	テキスト	⇒	ファイル名
attachCount	整数	⇐	添付の数
戻り値	整数	⇒	エラーコード

説明

ファイルに添付が含まれる場合、*MSG_HasAttach* コマンドは*attachCount*に添付数を返します。添付は非テキストMIMEの同封物です。メッセージに添付が含まれない場合、0が返されます。

fileName は、ヘッダ情報を取得するファイルの名前またはファイルのフルパスです。ファイル名のみを指定した場合、*POP3_SetPrefs* または *MSG_SetPrefs* で指定したパスでファイルを検索します (互換性メモ参照)。フォルダが指定されていない場合、ストラクチャと同階層 (4D シングルユーザ) または4Dクライアントフォルダ (4D Server) でファイルを探します。

互換性メモ (バージョン6.8.1): *MSG_SetPrefs* コマンドを使用しない場合、*POP3_SetPrefs*の*msgFolder* 引数が使用されます。*MSG_SetPrefs*が使用されていれば*POP3_SetPrefs*の*msgFolder* 引数は無視されます。

*attachCount*には、*fileName*に含まれる添付数が返されます。

MSG_MessageSize

MSG_MessageSize (fileName ; headerSize ; bodySize ; msgSize) -> 戻り値

引数	型		説明
fileName	テキスト	⇒	ファイル名
headerSize	倍長整数	⇐	ヘッダサイズ (環境設定がONであればLFを取り除く)
bodySize	倍長整数	⇐	ボディサイズ (環境設定がONであればLFを取り除く)
msgSize	倍長整数	⇐	メッセージ全体またはファイルサイズ (環境設定を無視)
戻り値	整数	↩	エラーコード

説明

MSG_MessageSize コマンドは、*POP3_Download* コマンドでディスクに保存したメッセージファイルのファイル名を指定して、メッセージの様々な部分のサイズを返します。

fileName は、ヘッダ情報を取得するファイルの名前またはファイルのフルパスです。ファイル名のみを指定した場合、*POP3_SetPrefs* または *MSG_SetPrefs* で指定したパスでファイルを検索します (互換性メモ参照)。フォルダが指定されていない場合、ストラクチャと同階層 (4D シングルユーザ) または4Dクライアントフォルダ (4D Server) でファイルを探します。

headerSize 倍長整数変数にはヘッダのサイズが返されます。

bodySize 倍長整数変数にはボディのサイズが返されます。

これら二つの引数は、*POP3_SetPrefs* や *MSG_SetPrefs*の*stripLineFeed* 引数を考慮に入れます。

互換性メモ (バージョン6.8.1): *MSG_SetPrefs* コマンドを使用しない場合、*POP3_SetPrefs*の*msgFolder*と*stripLineFeed* 引数が考慮されます。*MSG_SetPrefs*が使用されていれば*POP3_SetPrefs*の*msgFolder*と*stripLineFeed*引数は無視されます。

*msgSize*にはメッセージ全体のサイズが返されます。

MSG_SetPrefs

MSG_SetPrefs (stripLineFeed ; msgFolder ; attachFolder) -> 戻り値

引数	型		説明
stripLineFeed	整数	→	0 = LFを保持する, 1 = LFを取り除く, -1 = 変更しない
msgFolder	テキスト	→	メッセージフォルダパス ("" = 変更しない)
attachFolder	テキスト	→	添付フォルダパス ("" = 変更しない)
戻り値	整数	←	エラーコード

説明

MSG_SetPrefs コマンドは、MSGコマンドの環境を設定します。

stripLineFeed は、ダウンロードしたメッセージでラインフィードをどのように扱うかを指定する整数値です。ほとんどのインターネットメッセージでは行の終了を示すために CR/LFの組を使用しています。Macintoshアプリケーションは改行コードとしてキャリッジリターンのみを使用します。このオプションを使用してメッセージテキストからラインフィード文字を取り除くことができます。0を設定すると、メッセージはそのままになります。1を指定すると、メッセージからラインフィードが取り除かれます。-1は以前の設定を変更しないことを意味します。デフォルト値は1で、メッセージのラインフィードは取り除かれます。

msgFolder は、取得したメッセージが格納されているフォルダのローカルパスです。

互換性メモ (version 6.8.1): *MSG_SetPrefs* コマンドが使用されていない場合、*POP3_SetPrefs* の*stripLineFeed*と*msgFolder*引数が考慮されます。*MSG_SetPrefs* コマンドが使用されると、*POP3_SetPrefs*の引数は無視されます。

attachFolder は添付を格納するフォルダのパスで、*MSG_Extract* コマンドがメッセージから添付を取り出す際、このフォルダに格納します。

互換性メモ (version 6.8.1): この引数は*POP3_SetPrefs*と*MSG_SetPrefs*に存在します。これらいずれかのコマンドを使用してこの設定を変更できます。

MSG_SetPrefs コマンドの利用を強く推奨します。*POP3_SetPrefs* 引数は互換性のために残されており、将来使われなくなります。*POP3_SetPrefs* コマンドの*attachFolder* 引数はオプションです。この引数を使用しないことを強くお勧めします。この推奨は*POP3_GetPrefs*にも当てはまります。

IC ファイル転送

- ファイル転送 - 概要
- FTP_Append
- FTP_ChangeDir
- FTP_Delete
- FTP_GetDirList
- FTP_GetFileInfo
- FTP_GetPassive
- FTP_GetType
- FTP_Login
- FTP_Logout
- FTP_MacBinary
- FTP_MakeDir
- FTP_PrintDir
- FTP_Progress
- FTP_Receive
- FTP_RemoveDir
- FTP_Rename
- FTP_Send
- FTP_SetPassive
- FTP_SetType
- FTP_System
- FTP_VerifyID

ファイル転送 - 概要

File Transfer Protocol (FTP) は第一義的にコンピュータから他のコンピュータにドキュメントやファイルを転送するためのプロトコルです。FTPサイトは、FTPサーバソフトウェアを走 らせているコンピュータです。File Transfer Protocolは異機種間でのファイル交換の方法を提供します。さまざまなプラットフォーム上のクライアントアプリケーションがFTPサーバにログイン して、テキストやバイナリファイルをアップロードしたりダウンロードしたりできます。4D Internet CommandsのFTPルーチンを使用して、4Dデータベースで FTPクライアントを作成できます。

Notes:

- FTPコマンドでパス名を指定する際、FTPサイト上のファイルの場所は常にUnixディレクトリとして扱います。たとえFTPサーバソフトウェ アがMacintosh上で実行されているとしてもです。プラットフォームにかかわらず、FTPサーバはUnixパス名を、内部的に変換します。
- 柔軟性を高めるため、4D Internet Commandsでは、POP3, IMAP, FTP接続参照を直接低レベルTCPコマンドに渡すことを、あるいはその逆を許可しています。詳細は**低レベルルーチン - 概要**を参照してください。

FTP_Append

FTP_Append (ftp_ID ; localPath ; hostPath ; progress) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	⇒	FTPログイン参照
localPath	テキスト	⇒	送信するドキュメントのパス名
hostPath	テキスト	⇒	ドキュメントの送信先パス名
progress	整数	⇒	1 = 進捗表示, 0 = 進捗を隠す
戻り値	整数	⇐	エラーコード

説明

FTP_Append コマンドは、*hostPath* 引数で指定した既存のファイルの最後にデータを追加すること以外、*FTP_Send*と同じ動作をします。このコマンドの主な機能は、既存のテキストファイルの最後にデータを追加することです。

FTP_ChangeDir

FTP_ChangeDir (ftp_ID ; directory) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	➡	FTPログイン参照
directory	テキスト	➡	Unixディレクトリパス名
戻り値	整数	⮌	エラーコード

説明

FTP_ChangeDir コマンドはカレントのワーキングディレクトリ (CWD) を*directory* 引数で与えられたパスに変更します。

*FTP_GetDirList*と*FTP_GetFileInfo*コマンドもカレントのワーキングディレクトリを変更します。しかし*FTP_ChangeDir* コマンドの実行はより速く、引数の数も少ないです。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

directory は既存のFTPディレクトリを参照するHostPathフォーマットのテキスト値です。ディレクトリが存在しないか、操作を行うためのアクセス権がない場合、エラーが返されます。この場合カレントワーキングディレクトリは変更されません。

例題

このコードはCWDをFTPルートに設定します::

```
$err:=FTP_ChangeDir(ftp_ID;"/")
```

FTP_Delete

FTP_Delete (ftp_ID ; hostPath) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	⇒	FTPログイン参照
hostPath	テキスト	⇒	ドキュメントへのパス名
戻り値	整数	⇐	エラーコード

説明

HostPathフォーマットでファイルのパス名を渡すと、*FTP_Delete* コマンドはFTPサーバから指定したファイルを削除します。この操作を行うためのアクセス権がない場合、エラーが返されます。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

hostPath は削除するドキュメントのパスです。*hostPath* 引数の値はフルパス名またはファイル名です。短い名前を渡す場合、指定したファイルはCWDの中になければなりません。

Note: *FTP_ChangeDir* コマンドを使用してCWDを変更できます。*FTP_PrintDir* コマンドを使用すればCWDを取得できます。

FTP_GetDirList

FTP_GetDirList (ftp_ID ; directory ; names ; sizes ; kinds ; modDates ; modTimes) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	⇒	FTPログイン参照
directory	テキスト	⇒	Unixディレクトリパス名
		⇐	カレントディレクトリ
names	文字配列	⇐	名前のリスト
sizes	倍長整数配列	⇐	サイズのリスト
kinds	整数配列	⇐	種類のリスト: 0 = プレーンファイル, 1 = ディレクトリ, 2 = ブロックタイプ特別ファイル, 3 = 文字タイプ特別ファイル, 4 = シンボリックリンク, 5 = FIFO 特別ファイル, 6 = AF_UNIX アドレスファミリソケット
modDates	日付配列	⇐	更新日のリスト
modTimes	倍長整数配列	⇐	更新時間のリスト
戻り値	整数	↩	エラーコード

説明

FTP_GetDirList コマンドは、*ftp_ID*で指定されたFTPセッションで、*directory*内のオブジェクトリストを取得します。*directory*中の項目の名前、サイズ、タイプ、更新日、そしてオプションで更新時間に関する情報が配列に返されます。FTPサイトへの接続が*FTP_Login*によって開かれ、まだ有効でなければなりません (*FTP_VerifyID*)。*FTP_GetDirList* コマンドはカレントのワーキングディレクトリ (CWD) を指定されたパスに変更し、*directory* 引数に返します。

*ftp_ID*は、*FTP_Login*により確立されたFTPセッション参照です。

directory はFTPディレクトリを参照するHostPath フォーマットのテキスト値です。変更後のカレントディレクトリを受け取るため、4D変数またはフィールドをこの引数に渡さなければなりません。通常この引数に返される値は、渡した値と同じです。しかし、アクセス制限などの理由で、ディレクトリの変更が行われない場合があります。この場合、*directory* 引数はセッションのカレントディレクトリへのHostPath を保持します。

空の文字列をこの引数に渡すと、カレントディレクトリのファイルリストが配列に返され、カレントディレクトリのHostPath が*directory* 引数に返されます。

names には文字列またはテキスト配列を渡し、*directory*で指定したパスのオブジェクト名が返されます。

sizes には倍長整数配列を渡し、*directory*で指定したパスのオブジェクトのサイズが返されます。

*kinds*には整数配列を渡し、*directory*で指定したパスのオブジェクトの種類が返されます。それぞれの数値は以下のオブジェクトを指します:

kind	ファイルタイプ
0	plain file
1	directory
2	block-type special file
3	character-type special file
4	symbolic link (aliases on files or folders)
5	FIFO special file
6	AF_UNIX address family socket

Note: symbolic link (種類=4)の場合、FTPサーバは特殊なパス名を返します (エイリアス名 + シンボル + ソースファイルまたはフォルダへのパス)。このパス名を使用してソースファイルやフォルダにアクセスしようとすると、エラーが返されます。*FTP_GetDirList* から返された文字列の、シンボルより後ろの文字列から、ソースファイルやフォルダのパス名を展開しなくてはなりません。そうしないと、*FTP_GetFileInfo*などのコマンドはファイルやフォルダを見つけられないため、エラー-10085を返します。

modDates には日付配列を渡し、*directory*で指定したパスの最終更新日が返されます。

modTimes には倍長整数配列を渡し、*directory*で指定したパスの最終更新時間が返されます。

注意: 4Dでは時間タイプのデータを扱うために倍長整数配列を使用します。それぞれの配列要素は秒数を表します。これらの値をHH:MM:SSフォーマットに変換するには、**Time string**コマンドを使用します。

FTP_GetFileInfo

FTP_GetFileInfo (ftp_ID ; hostPath ; size ; modDate ; modTime) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	⇒	FTPログイン参照
hostPath	テキスト	⇒	ドキュメントへのパス名
size	倍長整数	⇐	ドキュメントサイズ
modDate	日付	⇐	更新日
modTime	時間	⇐	更新時間
戻り値	整数	⇐	エラーコード

説明

FTP_GetFileInfo コマンドは、*hostPath*で指定されたファイルの最終更新に関する情報を返します。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

hostPath は情報を取得するドキュメントへのパスです。

Note: *FTP_GetFileInfo* コマンドは、*hostPath* がフルパス名で現在のカレントワーキングディレクトリ (CWD) と異なるディレクトリを示す場合、CWDを変更することがあります。この場合、CWDは*hostPath* 引数で指定したディレクトリとなります。

size には*hostPath*で指定したファイルのサイズが返されます。

modDate と *modTime* にはファイルの最終更新日と時間が返されます。

FTP_GetPassive

FTP_GetPassive (ftp_ID ; passiveMode) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	⇒	FTPログイン参照
passiveMode	整数	⇐	現在のデータストリーム転送モード: 0=Activeモード, 1=Passiveモード
戻り値	整数	⇒	エラーコード

説明

FTP_GetPassive コマンドは現在のデータストリーム転送モードを返します。

FTP転送モードについては、*FTP_SetPassive* コマンドの説明を参照してください。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

passiveMode には現在のデータストリーム転送モードが返されます。:

- 0が返された場合、FTPサーバにActiveモードで接続します。
- 1が返された場合、FTPサーバにPassiveモードで接続します。(デフォルト)

FTP_GetType

FTP_GetType (ftp_ID ; ftp_Mode) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	⇒	FTPログイン参照
ftp_Mode	文字	⇐	"A" = Ascii; "I" = イメージ; "L 8" = Logical 8-bit
戻り値	整数	⇒	エラーコード

説明

FTP_GetType コマンドは現在のFTP転送に関する情報を返します。転送モードは*FTP_SetType* コマンドで設定できます。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

ftp_Mode には現在のFTP転送モードを示すコードが返されます。

FTP_Login

FTP_Login (hostName ; userName ; password ; ftp_ID ; welcomeText) -> 戻り値

引数	型		説明
hostName	文字	⇒	ホスト名またはIPアドレス
userName	文字	⇒	ユーザ名
password	文字	⇒	パスワード
ftp_ID	倍長整数	⇐	FTPセッション参照
welcomeText	テキスト	⇐	FTP Welcomeテキスト
戻り値	整数	⇒	エラーコード

説明

FTP_Login コマンドは、*hostName*で指定したFTPサーバとの接続を確立し、*userName*と*password*を使用してシステムにログインします。

hostName はリモートシステムのホスト名またはIPアドレスです。

userName はFTPサーバが認証に使用するユーザアカウント名です。多くのFTPサーバは"anonymous"ユーザ名によるゲストアクセスをサポートしています。anonymousでログインする際は、慣習的に電子メールアドレスをパスワードとして渡します。

password はシステム上の*userName* に対応するパスワードです。

ftp_ID には新しく開かれたセッションの参照が返されます。この値は続くFTPコマンドで使用されます。この引数には4Dの変数またはフィールドを渡します。

welcomeText はオプションの引数で、弓座ーがシステムにログインした際に返されるテキストが格納されます。多くのFTPサイトで、ログイン時にWelcomeメッセージが表示されます。この引数を渡す場合、4Dの変数またはフィールドを渡します。

例題

```
$OK:=False
Case of
   :(FTP_Login("ftp.4d.com";"anonymous";"dbody@aol.com";vFTP_ID;vFTP_Msg)#0)
   :(FTP_Progress(-1;-1;"Progress window";"Getting requested file…";"*")#0)
   :(FTP_Send(vFTP_ID;"My Hard Drive:Documents ƒ:July Sales Report";"/pub/reports";1)#0)
   :(FTP_Logout(vFTP_ID)#0)
   Else
      $OK:=True  `すべてのコマンドがエラーなしで実行された
End case
```

Note: Case of のこの特別な使い方については、**Appendix A, プログラムTips**を参照してください。

FTP_Logout

FTP_Logout (ftp_ID) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	⇒	FTPログイン参照
		⇐	0 = セッションをクローズした
戻り値	整数	↩	エラーコード

説明

開かれたFTPセッション参照を渡すと、*FTP_Logout* コマンドはサーバを切断し、セッションが使用していたメモリを解放します。このコマンドは、セッションのクローズに成功すると*ftp_ID* 引数に0を返します。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

例題

```
If(FTP_Login("ftp.4d.com";"anonymous";vEmailID;vFTP_ID;vFTP_Msg)=1)
   $error:=FTP_Send(vFTP_ID;"My Hard Drive:Documents:Sales Report";"/pub/reports";1)
   $error:=FTP_Logout(vFTP_ID)
End if
```

FTP_MacBinary

FTP_MacBinary (ftp_ID ; macBinaryMode) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	⇒	FTPログイン参照
macBinaryMode	整数	⇒	-1 = 現在の設定を取得, 1 = 有効, 0 = 無効
		⇐	現在の設定 (-1 を渡した場合)
戻り値	整数	↩	エラーコード

説明

FTP_MacBinary コマンドは、*FTP_Send* や *FTP_Receive*使用時のMacBinaryモードを有効/無効にします。*ftp_ID*でカレントのFTPセッションを特定すると、このコマンドは*macBinaryMode* 引数に渡した値に基づき、MacBinary転送をオンまたはオフにします。

MacBinaryプロトコルはしばしば、データフォークとリソースフォークを持つバイナリデータやファイルの転送をおこなうために、Macintosh FTPクライアントとサーバにより使用されます。

Windowsユーザのためのメモ: FTP転送時にWindows上でMacBinaryプロトコルを使用できますが、PCコンピュータ上でMacBinaryファイルをデコードすることには意味がありません。Intelベースのマシンはデータフォークとリソースフォークを持つデータを格納できません。PCプラットフォームでは、このようなファイルを正しく認識できないため、リソースフォークを持つMacintoshファイルはエンコードされていないフォーマットで保存する際壊れてしまうでしょう。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

macBinaryMode はMacBinary転送をオンにするかオフにしるか指定するための整数値です。コマンドから変更後のMacBinary転送モードが返されるため、この引数は変数であるべきです。1を渡すとMacBinaryが有効になり、0を渡すと無効になります。-1を渡すと、コマンドは*macBinaryMode* 引数に現在のMacBinary転送モードを返します (1 または 0)。

警告： すべてのFTPサーバがMacBinaryプロトコルをサポートするわけではありません。このような場合、*macBinaryMode* 引数の値にかかわらず、*FTP_MacBinary*コマンドをコールするとエラー10053が返されます。

例題

この例題では、FTPファイルを受信する前に、MacBinaryプロトコルを有効にします。MacBinaryが有効な状態でファイルの受信に成功すると、オリジナルのフォーマットにデコードされ、MacBinaryドキュメントは削除されます。

```
vUseMacBin:=-1
$error:=FTP_MacBinary(vFTP_ID;vUseMacBin)
If($error=10053)
   MacBinaryIsSupported:=False `FTPサーバはMacBinaryプロトコルをサポートしない
Else
   MacBinaryIsSupported:=True
End if



vLocalFile:=""
If(MacBinaryIsSupported)
   vUseMacBin:=1
   $error:=FTP_MacBinary(vFTP_ID;vUseMacBin) `ダウンロードのためにMacBinaryを有効にする
End if
$error:=FTP_Receive(vFTP_ID;"MyApplication";vLocalFile;cbShowTherm)
```

```
If($error=0) & (vUseMacBin=1) `受信ができ、ファイルがMacBinaryフォーマットなら
   vDecodePath:=""
   If(IT_Decode(vLocalFile;vDecodePath;8)=0) `MacBinary デコード
      DELETE DOCUMENT(vLocalFile) `ソースのデコードができたら、ファイルを削除
   End if
End if
```

FTP_MakeDir

FTP_MakeDir (ftp_ID ; directory) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	→	FTPログイン参照
directory	テキスト	→	Unixディレクトリパス名
戻り値	整数	↩	エラーコード

説明

有効な*directory*名を渡すと、*FTP_MakeDir* コマンドはカレントワーキングディレクトリ (CWD) に新しいフォルダディレクトリを作成します。この操作を行うためのアクセス権がない場合、エラーが返されます。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

directory はFTPディレクトリを参照するHostPathフォーマットのテキスト値です。*directory* 引数の値はフルパス名またはフォルダ名です。短い名前を渡すと、CWDにフォルダを作成します。*directory*には空白を含めないことをお勧めします。

Note: *FTP_ChangeDir* コマンドを使用してCWDを変更できます。*FTP_PrintDir* コマンドを使用すればCWDを取得できます。

FTP_PrintDir

FTP_PrintDir (ftp_ID ; directory) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	➡	FTPログイン参照
directory	テキスト	⬅	Unixディレクトリパス名
戻り値	整数	➲	エラーコード

説明

FTP_PrintDir コマンドは*directory* 引数にカレントワーキングディレクトリ (CWD) を返します。

*FTP_GetDirList*コマンドもカレントワーキングディレクトリを返します。しかし*FTP_PrintDir* コマンドの実行は速く、引数も少ないです。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

directory 引数にはカレントワーキングディレクトリが返されます。

例題

この例題では、$Cwd変数にカレントワーキングディレクトリが返されます。

```
$err:=FTP_PrintDir(ftp_ID;$Cwd)
```

FTP_Progress

FTP_Progress (left ; top ; windowTitle ; thermoText ; cancel) -> 戻り値

引数	型		説明
left	整数	➡	ウィンドウ左座標
top	整数	➡	ウィンドウ上座標
windowTitle	文字	➡	サーモメータウィンドウタイトル
thermoText	文字	➡	サーモメータの進捗文字列
cancel	文字	➡	キャンセルボタンラベル
戻り値	整数	↩	エラーコード

説明

FTP_Progress コマンドは、FTP進捗インジケータウィンドウの座標とダイアログボックステキストを指定します。進捗インジケータは*FTP_Send*, *FTP_Append*, *FTP_Receive*の実行中に表示されます。*FTP_Progress* コマンド自身は進捗ウィンドウを表示しません。このコマンドは送受信コマンド実行時に表示されるウィンドウの設定を行うだけです。*FTP_Send*, *FTP_Append*, *FTP_Receive*いずれのコマンドも、進捗ウィンドウの表示/非表示を設定する引数を持っています。

進捗ウィンドウはファイル転送終了時に自動で閉じられます。何らかの理由で送受信するファイルのサイズを決定できない場合、バーバータイプのサーモメータが表示され、ファイルサイズは"unknown"と表示されます。

*left*はサーモメータ進捗ウィンドウの左座標です。*left*が-1の場合、ウィンドウは水平方向中央に配置されます。

*top*はサーモメータ進捗ウィンドウの上座標です。*top*が-1の場合、ウィンドウは垂直方向中央に配置されます。

windowTitle はサーモメータ進捗ウィンドウのタイトルです。以下の例題ではウィンドウタイトルに"Getting '/pub/CGMiniViewer.hqx'"を設定しています。*windowTitle*が空の文字列の場合、ウィンドウにはタイトルが付けられません。

thermoText は進捗サーモメータの上部に表示されるテキストです。*thermoText*に"*"を指定すると、デフォルトのテキストが使用されます。以下の例題では、*thermoText*に"Receiving File: /pub/CGMiniViewer.hqx"が指定されています。サーモメータのデフォルトテキストは、ホストから送信される転送プロセスのステータステキストです。このテキストは接続が次の転送プロセスステージに進むと変更されます。

cancel はCancelボタンのテキストです。*cancel*が空の文字列の場合、Cancelボタンは隠されます。*cancel*に"*"を指定すると、デフォルトの"Cancel"が使用されます。

例題

```
$error:=FTP_Progress(-1;-1;"Getting '/pub/CGMiniViewer.hqx'";"*";"*")
Case of
   :(FTP_Login("ftp.4d.com";"anonymous";"dbody@aol.com";vFTP_ID;vFTP_Msg)#0)
   :(FTP_Receive(vFTP_ID;"/pub/CGMiniViewer.hqx";"HardDrive:Docsf:4D";1)#0)
   :(FTP_Logout(vFTP_ID)#0)
   Else
      $OK:=True  `すべてのコマンドがエラーなしで実行された
```

```
    End case
```

Note: この特別な **Case of** 構造の利用法については、**Appendix A, プログラムTips**を参照してください。

FTP_Receive

FTP_Receive (ftp_ID ; hostPath ; localPath ; progress) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	⇒	FTPログイン参照
hostPath	テキスト	⇒	受信するドキュメントのパス名
localPath	テキスト	⇒	ドキュメントの保存先パス名
		⇐	結果ファイルパス名 (""が渡された場合)
progress	整数	⇒	0 = 進捗を隠す, 1 = 進捗を表示する
戻り値	整数	⇐	エラーコード

説明

FTP_Receive コマンドはFile Transfer Protocolを使用して、*hostPath*で指定した場所からファイルを受信します。*FTP_Receive*は保存先に既にドキュメントが存在する場合、エラー-48を返します。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

hostPath には受信するドキュメントのパスを指定します。*hostPath*がドキュメントへのフルパスでない場合、コマンドはエラーを返します。パスはUnix形式で指定し、スラッシュ ("/") で区切ります。詳細は**用語解説**の節を参照してください。

localPath はドキュメントの保存先を指定するテキスト値です。*localPath*が空の文字列の場合、標準の保存ダイアログが表示され、結果のパス名が*localPath*変数に返されます。*localPath*がファイル名の場合、ファイルはストラクチャと同階層 (4Dシングルユーザ) または4Dクライアントフォルダ (4D Server) に保存されます。パスはローカルドキュメントのため、パス名はプラットフォームに対応した区切り文字で区切ります。詳細は**用語解説**の節を参照してください。

progress は進捗インジケータを表示するかしないかを指定する整数値です。1を渡すと進捗インジケータを表示します。0を渡すと隠します。

例題

```
vUseMacBin:=-1
$error:=FTP_MacBinary(vFTP_ID;vUseMacBin)
If($error=10053)
   MacBinaryIsSupported:=False `FTPサーバはMacBinaryプロトコルをサポートしていない
Else
   MacBinaryIsSupported:=True
End if

vLocalFile:=""
If(MacBinaryIsSupported)
   vUseMacBin:=1
   $error:=FTP_MacBinary(vFTP_ID;vUseMacBin) `ダウンロード用にMacBinaryを有効にする
   $error:=FTP_Receive(vFTP_ID;"CGMiniViewer.hqx";vLocalFile;cbShowTherm)
   If($error=0) & (vUseMacBin=1)
      vDecodePath:=""
      If(IT_Decode(vLocalFile;vDecodePath;8)=0) `MacBinary デコード
         DELETE DOCUMENT(vLocalFile) `ソースのデコードができたらファイルを削除する
      End if
   End if
End if
```

FTP_RemoveDir

FTP_RemoveDir (ftp_ID ; directory) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	➡	FTPログイン参照
directory	テキスト	➡	Unixディレクトリパス名
戻り値	整数	⮌	エラーコード

説明

有効な*directory*名を渡すと、*FTP_RemoveDir* コマンドはフォルダディレクトリを削除します。この操作を行うためのアクセス権がない場合、エラーが返されます。さらに、項目を含むディレクトリを削除しようとすると、おそらくセキュリティエラーが発生します。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

directory は既存のFTPディレクトリを参照するHostPathフォーマットのテキスト値です。*directory* 引数の値はフルパス名またはフォルダ名です。短い名前を渡す場合、指定したフォルダはCWD内になければなりません。

Note: *FTP_ChangeDir* コマンドを使用してCWDを変更できます。*FTP_PrintDir* コマンドを使用すればCWDを取得できます。

FTP_Rename

FTP_Rename (ftp_ID ; hostPath ; newName) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	→	FTPログイン参照
hostPath	テキスト	→	FTPサーバ上のドキュメントパス名
newName	テキスト	→	新しいドキュメント名
戻り値	整数	↩	エラーコード

説明

*hostPath*フォーマットでファイルのパス名を渡すと、*FTP_Rename* コマンドはFTPサーバ上の指定したファイルの名称を変更します。この操作を行うためのアクセス権がない場合、エラーが返されます。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

hostPath は名称変更するドキュメントへのパス名です。*hostPath* 引数の値はフルパス名またはファイル名です。短い名前を渡す場合、指定したファイルはCWDの中になければなりません。

newPathName には新しいドキュメントのパス名を渡します。

Note: *FTP_ChangeDir* コマンドを使用してCWDを変更できます。*FTP_PrintDir* コマンドを使用すればCWDを取得できます。

FTP_Send

FTP_Send (ftp_ID ; localPath ; hostPath ; progress) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	➡	FTPログイン参照
localPath	テキスト	➡	送信するドキュメントのパス名
hostPath	テキスト	➡	ドキュメントの送信先パス名
progress	整数	➡	1 = 進捗表示, 0 = 進捗を隠す
戻り値	整数	↩	エラーコード

説明

FTP_Send コマンドはFTPセッション参照、送信するドキュメントのパス、送信先のパスを受け取ると、リモートマシンにドキュメントを送信します。*FTP_Send* はFTPファイルステータスエラーが返されると即座に制御を返します。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

localPath は送信するドキュメントのパスです。*localPath*が空の文字列の場合、標準のファイルを開くダイアログボックスが表示されます。*localPath*がパスを含まないファイル名の場合、コマンドはストラクチャと同階層 (4Dシングルユーザの場合) または4Dクライアントフォルダ (4D Server) でファイルを探します。パスはローカルドキュメントを指すので、ディレクトリはプラットフォームに対応した文字で区切らなくてはなりません。詳細はこのマニュアルの**用語解説**を参照してください。

hostPath はファイル名を含むドキュメントの送信先パスです。*hostPath*は、FTPサーバがファイルを受信後の、ファイル名を表します。*localPath*が空の文字列の場合ディスク上のファイル名が使用されるので、*hostPath*も空の文字列にできます。選択したファイル名が使用されます。

*hostPath*にはフルパス名またはファイル名を指定できます。フルパス名を指定すると、ファイルは*hostPath*で指定したディレクトリに置かれます。ファイル名を渡すか、ファイル選択に空の文字列を使用した場合、ファイルはカレントワーキングディレクトリ (CWD) に送信されます。

ファイル名やファイルパスが正しく解決できない場合、コマンドはエラーを返します。ユーザがディレクトリにファイルを送信する権限を持たない場合、エラーが返されます。パスはスラッシュ ("/") で区切ります。詳細はこのマニュアルの**用語解説**を参照してください。

progress は進捗インジケータを表示するかしないかを指定する整数値です。1を渡すと進捗インジケータを表示します。0を渡すと隠します。

例題 1

```
$OK:=False
Case of
   :(FTP_Login("ftp.4d.com";"anonymous";vEmailID;vFTP_ID;vFTP_Msg)#0)
   :(FTP_Progress(-1;-1;"Progress window";"Getting requested file…";"Cancel")#0)
   :(FTP_Send(vFTP_ID;"My Hard Drive:Documents:July Sales Report";"/pub/reports/";1)#0)
   :(FTP_Logout(vFTP_ID)#0)
   Else
      $OK:=True  `すべてのコマンドがエラーなしで実行された
End case
```

Note: この特別な **Case of** 構造の利用法については、**Appendix A, プログラムTips**を参照してください。

例題 2

```
$error:=FTP_Send(vFTP_ID;"";"";1)
```

FTP_SetPassive

FTP_SetPassive (ftp_ID ; passiveMode) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	⇒	FTPログイン参照
passiveMode	整数	⇒	0=Activeモード, 1=Passiveモード (デフォルト)
戻り値	整数	⇐	エラーコード

説明

FTP_SetPassive コマンドは、*FTP_GetDirList*, *FTP_Send*, *FTP_Append*, *FTP_Receive*などのコマンドを実行する際の、FTPサーバとクライアント間のデータストリーム転送モードを設定します。データストリーム転送モード設定は、*FTP_SetPassive*が一度実行されるとこれらのコマンドに適用されます。

FTPサーバとクライアント間の転送は、二つのストリームに基づき行われます: コントロールストリーム (デフォルトでポート21) とデータストリーム (デフォルトでポート20)。通常、FTPサーバはデータ接続を開いて管理するために、"active"に定義されています。

歴史的な理由から、4D Internet CommandsはPassiveモードでFTPサーバへのデータストリーム転送モードを開こうとします。つまりデータストリームの交換を開始する前に、FTP コマンド "PASV"が送信されます。

しかし、いくつかのFTPサーバはPassiveモードをサポートせず、またファイアウォールが禁止している場合もあります。このような場合、カレントのデータストリーム転送モードをActiveにする必要があります。

Note: FTPで使用するモードをPassiveにするかActiveにするか、ネットワーク管理者に問い合わせる必要があります。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

passiveMode 引数はデータストリーム転送モードを指定します:

- 0を指定すると、FTPサーバにActiveモードで通信を行います。
- 1を指定すると、FTPサーバにPassiveモードで通信を行います。(デフォルト)

FTP_SetType

FTP_SetType (ftp_ID ; ftp_Mode) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	⇒	FTPログイン参照
ftp_Mode	文字	⇒	"A" = Ascii; "I" = [デフォルト] イメージ; "L 8" = Logical 8-bit
戻り値	整数	⇐	エラーコード

説明

FTP_SetType コマンドは送受信時に使用するFTP転送モードを変更するために使用します。この設定をデフォルトから変更する必要は特にありません。しかし、さまざまなプラットフォーム間やFTP実装の違いから、特定のタイプのFTP転送で変更が必要になる場合があるかもしれません。特に、プレインテキストドキュメントの転送時には、テキストファイルを正しく転送するために、Asciiモードに変更する必要があるかもしれません。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

ftp_Mode には、以降の送受信操作で使用する転送モードを指定するためのコードを渡します。デフォルトでImage ("I") 転送モードが使用されます。

FTP_System

FTP_System (ftp_ID ; systemInfo) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	⇒	FTPログイン参照
systemInfo	文字	⇐	システム情報
戻り値	整数	⇒	エラーコード

説明

FTP_System コマンドは*systemInfo*にFTPサーバソフトウェアに関する情報を返します。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

systemInfo にはFTPサーバに関する情報が返されます。

FTP_VerifyID

FTP_VerifyID (ftp_ID) -> 戻り値

引数	型		説明
ftp_ID	倍長整数	⇒	FTPログイン参照
		⇐	0 = 接続が既に閉じられている
戻り値	整数	⇒	エラーコード

説明

FTPサーバは、管理者によって設定された時間の間接続がないアカウントを切断します。FTPサーバと通信を行うコマンドはこのタイマをリセットします。*FTP_VerifyID* コマンドは指定したFTPセッションのタイマを、現在の状況やディレクトリを変更することなくリセットします。これにより、セッションがタイムアウトするかもしれない状況でも、セッションを有効に保つことができます。

FTP_VerifyID コマンドを実行すると、接続が既に閉じられているか検証します。セッションがまだ開かれていれば、コマンドはFTPサーバにセッションタイムアウトカウンタをリセットするよう指示します。セッションが既に閉じられていれば、*FTP_VerifyID*は対応するエラーを返し、FTPセッションで使用されていたメモリを解放し、*ftp_ID*に0を返します。

ftp_ID は、*FTP_Login*により確立されたFTPセッション参照です。

IC メール送信

- メール送信 - 概要
- SMTP_AddHeader
- SMTP_Attachment
- SMTP_Auth
- SMTP_Bcc
- SMTP_Body
- SMTP_Cc
- SMTP_Charset
- SMTP_Clear
- SMTP_Comments
- SMTP_Date
- SMTP_Encrypted
- SMTP_From
- SMTP_GetPrefs
- SMTP_Host
- SMTP_InReplyTo
- SMTP_Keywords
- SMTP_New
- SMTP_QuickSend Updated 13.2
- SMTP_References
- SMTP_ReplyTo
- SMTP_Send Updated 13.2
- SMTP_Sender
- SMTP_SetPrefs
- SMTP_Subject
- SMTP_To

メール送信 - 概要

Simple Mail Transfer Protocol (SMTP) はインターネットで使用されるメール標準です。4D Internet Commandsを使用して、一つのコマンドで単純なメッセージを構築したり、一連のコマンドを使用して複雑なメッセージを構築したりできます。SMTP コマンドにより、Reply-To、Sender、Attachement、Comment、Referencesなどメッセージのすべての部分をコントロールできます。

4Dと4D Internet Commandsを使用して、インターネットにメッセージや添付ファイルを送信するとてもパワフルなデータベースを作成できます。SMTPコマンドを使用してデータベースを拡張できる例としては以下のようなものがあります:

- 4Dで作成されたレポートやドキュメントを自動送信する。
- データベースの特殊状況 (エラーなど) を開発者に通知する。
- メーリングリストへのメール送信。

SMTPコマンドの利用シーンは無限です。これらのコマンドと、POP3 (メッセージや添付ファイルの取得)、FTP、TCPなどを使用すれば、4D開発者はデータベースの通信機能を劇的に拡張できます。

メールメッセージを作成する二つの方法

SMTPコマンドでは、メールを送信する単純な方法と複合的な方法、二つの方法が提供されます。単純な方法は*SMTP_QuickSend*コマンドを使用して行われ、必要なすべての引数を一つのコマンドに渡します。

送信するメッセージの構造はほとんどの場合極めて単純です。"ここの"誰かが、とある"題材"について、"どこかの"誰かに、なんらかの"メッセージ"を送信するということです。これが紙の場合、あなたはそれを書き留め、封筒に入れたら住所を書いて郵便局に持っていきます。 *SMTP_QuickSend*では、これら"From"、"To"、"Subject"そして"Message Body"要素を一つのコマンドに渡すだけでメールが送信されます。

しかしすべてのメールがこのように簡単に構築できるわけではありません。たとえば他のグループにメールのコピーを送信しなければならない場合や、年次レポートを添付して送らなければならない場合があります。4D Internet CommandsのSMTPコマンドでは、メールのすべての要素をあなたがコントロールできます。複数の添付や、CC、BCCなどすべてのメールヘッダを SMTPコマンドで処理することができます。

メールの配送を理解する

SMTPコマンドを理解する上で理解すべき最も重要なことは、メールが受信者に配信される方法に関連することです。SMTPコマンドは受信者に直接メール を配信しません。コマンドはメールを正しい形に構築し、そののちに**SMTP_Host**コマンドで指定されたSMTPサーバにメールを送信します。SMTP サーバはしばしば組織自身の所有であるかまたはインターネットさビスプロバイダによって運営されています。SMTPサーバは最適化されたメールの配送経路 を解決し、管理者により指定されたスケジュールでメールを転送します。

複合SMTPメッセージを送信するために最低限必要なこと

SMTPコマンドで正しくメールを送信するためには、いくつかのコマンドを必ず使用してメールを定義しなければなりません。以下メール送信時に必ず使用しなければならないコマンドを示します:

- *SMTP_New*
  メモリ上に新しいメッセージのためのスペースを確保し、続くコマンドで使用するための参照を返します。
- *SMTP_Host*
  メッセージ転送を担当するSMTPを指定します。
- *SMTP_From*
  最低一つのアドレスが必要です。

- *SMTP_To*
  最低一つのアドレスが必要です。

- *SMTP_Send*
  メッセージを送信します。

- *SMTP_Clear*
  メッセージ作成時に使用したメモリをクリアします。

上にあげたコマンドのみを使用した場合、メッセージは件名と本文なしで送信されます。もちろんこれで用をなすわけではありません。有用なメッセージにするためには追加の情報を指定する必要があります。

SMTP_AddHeader

SMTP_AddHeader (smtp_ID ; headerName ; headerText ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	➡	メッセージ参照
headerName	文字	➡	ヘッダ名
headerText	テキスト	➡	ヘッダテキスト
deleteOption	整数	➡	0 = 追加, 1 = すべてのヘッダを'headerName'で置き換え, 2 = 'headerName'ヘッダをすべて取り除く
戻り値	整数	↩	エラーコード

説明

SMTP_AddHeader コマンドは、*smtp_ID*で参照されるメッセージに独自のヘッダを追加するために使用します。4D Internet Commandsが提供するさまざまなヘッダ関連のコマンドに加え、二つのカテゴリのヘッダ ('ユーザ定義' と '拡張') があります。**SMTP_AddHeade**r コマンドでは新しいヘッダタグと、それに割り当てるデータ両方の追加が可能です。

拡張ヘッダ: これらのヘッダはNICにより公式に認証されていて、オリジナルのSMTP使用の後に定義されました。これらのヘッダはしばしば、さまざまなアプリケーションの動作に影響する特定の機能があります。拡張ヘッダは"X"から始まっていてはなりません。

ユーザ定義ヘッダ: SMTPプロトコルは、独自のヘッダ定義を許可しています。すべてのユーザ定義ヘッダは"X-"から始まるべきで、このルールが守られる限り、将来の拡張ヘッダと衝突する心配はありません。ユーザ定義ヘッダは、通信の両端で開発者がコントロールを行えるようデザインをする際にとても便利です。

ユーザ定義ヘッダを使用して、開発者はデータをメッセージに格納し、*MSG_FindHeader* コマンドを使用して簡単にそのデータを取り出すことができます。たとえば"X-001001"という名前のヘッダを作成し、テーブル1のフィールド1の値を格納することができます。ユーザ定義ヘッダを使用すれば、情報を追加して、簡単にその情報を取り出すことができます。ボディ部を解析する必要はありません。

smtp_ID は*SMTP_New* コマンドで作成されるメッセージ参照です。

headerName は追加するヘッダのヘッダ名です。

headerText は*headerName*ヘッダに割り当てる情報を含むテキスト値です。

警告： テキストにラインフィード (ascii=10)を含んでいてはいけません。含まれていると、それはヘッダ部の終わりかつボディ部の始まりを意味します。これに続くヘッダはボディ部に押し出され、サーバやクライアントで正しく解釈されません。ヘッダについての詳細はRFC#822を参照してください。

deleteOption はオプションの引数で、カレントのヘッダを追加するか削除するかを指定します。0を指定すると、*headerName*をメッセージに追加します。1を指定すると、すべてのヘッダを*headerName*で置き換えます。この場合*headerName* が空の文字列なら、すべてのヘッダが取り除かれます。2を指定すると、*headerName*ヘッダを取り除きます。

例題

HTMLメールを送信するために、HTMLタグをメッセージボディに挿入し (例 <HTML>, <HEAD>, etc.) 、すべての "Content-Type" ヘッダを "text/html;charset=utf-8" で置き換えます:

```
If(Substring($body;1;6)="<HTML>")
   $err:=SMTP_AddHeader($SMTP_ID;"Content-Type:";"text/html;charset=utf-8";1)
End if
```

SMTP_Attachment

SMTP_Attachment (smtp_ID ; fileName ; encodeType ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	⇒	メッセージ参照
fileName	テキスト	⇒	添付するファイル名
encodeType	整数	⇒	0 = エンコードなし (データフォークのみ送信), +-1 = BinHex, +-2 = Base64; (データフォークのみ送信), +-3 = AppleSingle, +-4 = AppleDouble, +-5 = AppleSingle と Base64, +-6 = AppleDouble と Base64, +-7 = UUEncode
deleteOption	整数	⇒	0 = 既存のリストに追加, 1 = すべての添付をfileNameで置き換え, 2 = この添付を削除
戻り値	整数	⇐	エラーコード

説明

SMTP_Attachment コマンドは、MIMEフォーマットでテキストやバイナリファイルをメッセージに添付できるようにします。このコマンドは、一つのメールに複数のドキュメントを添付する際には複数回呼ばれます。0より大きな値が*encodeType* 引数に渡されると、このコマンドはメッセージの送信時にエンコードを実行します。

smtp_ID は*SMTP_New* コマンドで作成されるメッセージ参照です。

fileName はメッセージに添付するファイルを指定します。この値は以下三種の方法で指定できます:

""	= 標準のファイルを開くダイアログを表示する
"FileName"	= データベースストラクチャと同階層でファイルを検索する
"Path:FileName"	= ファイル名を含む完全なパス名

encodeType は、メッセージにファイルを含める際に適用するエンコーディングのタイプを指定する整数値です。添付ファイルがバイナリの場合、正しい変換がおこなわれるようエンコーディングメソッドを適用しなければなりません (BinHex, AppleSingle)。もっとも使用されるエンコーディングはBinHexです。

*encodeType*に正数を渡すと、コマンドはメッセージの送信時に自動でファイルをエンコードします。エンコーディングは*SMTP_Send* コマンドの実行時に行われます。ファイルのサイズが大きい場合、*SMTP_Send* コマンドの実行に時間がかかることになります。同じファイルを何度か送信する場合などは、*IT_Encode* コマンドでエンコーディングを行い、そのエンコード済みのファイルを *encodeType* に負数を渡して添付すると、時間を節約できます。*encodeType*に負数を渡すと、エンコーディングは行いませんが、メッセージヘッダに添付ファイルに関する正しいエンコーディング情報を設定します。受信者のメールアプリケーションはこの情報を使用して添付ファイルを解析する正しい方法を知ることができます。

Note: 配列要素を *encodeType* 引数に渡すことはできません。

deleteOption は添付をどのように扱うかを指定するオプションの整数引数です。0を指定すると、現在の添付リストに指定した添付ファイルを追加します。1を指定すると、すべての添付ファイルを添付ファイル*fileName*で置き換えます。*fileName* が空の文字列の場合、すべての添付ファイルが取り除かれます。2を指定すると、添付リストから*fileName*のみが取り除かれます。

SMTP_Auth

SMTP_Auth (smtp_ID ; userName ; password ; authMode) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	➡	メッセージ参照
userName	文字	➡	SMTP認証で使用するユーザ名
password	文字	➡	SMTP認証で使用するパスワード
authMode	整数	➡	使用する認証モード: 0 または省略 = サーバが指定するモード, 1= PLAIN, 2 = LOGIN, 3 = CRAM-MD5
戻り値	整数	⮌	エラーコード

説明

SMTP_Auth コマンドは、認証を要求するSMTPサーバを使用してメールを送信する場合に使用します。スパムなどの目的でメッセージや送信者の偽装リスクを軽減するために、SMTPサーバはこのような認証を要求します。

このコマンドは *userName* および *password*が空でない場合にのみ実行されるため、認証が必要であるかないかにかかわらず使用できます。

smtp_ID はSMTP_New コマンドで作成されるメールメッセージ参照です。

userName はSMTPサーバに認証のために送信されるユーザ名です。

password はSMTPサーバに認証のために送信されるパスワードです。

Note: *userName* または *password* が空の文字列の場合、*SMTP_Auth* コマンドは実行されません。

オプションの*authMode* 引数を使用して、使用する認証モードを強制することができます。

- 0を渡した場合、SMTP_Auth コマンドで使用される認証モードは、サーバがサポートするモードのうち最も安全性の高いモードとなります (CRAM-MD5, LOGIN そののち PLAIN)。
- 1を渡した場合、認証モードはPLAINが使用されます。
- 2を渡した場合、認証モードはLOGINが使用されます。
- 3を渡した場合、認証モードはCRAM-MD5が使用されます。

*authMode*が省略されると、デフォルトで0が使用されます。この引数で指定された認証モードがSMTPサーバでサポートされていない場合、エラーが返されます。

例題

この例題では、4Dデータベースに格納された特定のフィールドの内容に基づき、メッセージが認証ありまたはなしで送信されます:

```
C_INTEGER($vError)
C_LONGINT($vSmtp_id)
C_STRING(30;$vAuthUserName;$vAuthPassword)

$vError:=SMTP_New($vSmtp_id)
$vError:=SMTP_Host($vSmtp_id;"wkrp.com")
$vError:=SMTP_From($vSmtp_id;"herb_tarlick@wkrp.com")
$vError:=SMTP_Subject($vSmtp_id;"Are you there?")
$vError:=SMTP_To($vSmtp_id;"Dupont@wkrp.com")
$vError:=SMTP_Body($vSmtp_id;"Can we have a meeting?")
```

```
` サーバが認証を必要とする場合、以下のフィールドに値が
` 入力される。必要でなければ空のまま
$vAuthUserName:=[Account]AuthUser
$vAuthPassword:=[Account]AuthPass

$vError:=SMTP_Auth($vSmtp_id;$vAuthUserName;$vAuthPassword)
$vError:=SMTP_Send($vSmtp_id)
$vError:=SMTP_Clear($vSmtp_id)
```

SMTP_Bcc

SMTP_Bcc (smtp_ID ; blindCarbon ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	➡	メッセージ参照
blindCarbon	テキスト	➡	アドレスリスト
deleteOption	整数	➡	0 = 追加, 1 = 置き換え, 2 = 削除
戻り値	整数	⬅	エラーコード

説明

SMTP_Bcc コマンドは、*smtp_ID* で指定されるメッセージに、ブラインドカーボンコピーの受信者を追加します。Bcc: フィールドへの受信者の追加は必須ではありません。

グループにメールを送信する際、アドレスリストを他の受信者に表示させないようにする唯一の方法は、アドレスを"Bcc"ヘッダに記載することです。"Bcc"ヘッダにリストされたアドレスは、メッセージヘッダや本文には含まれません。"Bcc"に記載されたアドレスは、メールのすべての受信者に対して表示されません。

"Bcc"の受信者は"To" や "Cc" に記載された受信者を見ることができます。しかしすべての受信者は"Bcc"の受信者を見ることはありません。しばしばグループ宛へのメールでは、すべての受信者を"Bcc"ヘッダに記載して送信されます。これにより、受信者に膨大なアドレスリストを表示させたり、アドレスリストが他の受信者に漏れることを防ぐことができます。

"Bcc"を利用するもう一つのケースは、多くのメールアプリケーションに実装されている、"全員に返信"を使用されるケースです。この機能は"To" や "Cc"に記載されているすべての受信者を返信メールの宛先に設定するものです。すべての受信者を"Bcc"ヘッダに置くと、オリジナルメッセージの受信者すべてに返信が送信されることを防ぐことができます。

smtp_ID はSMTP_New コマンドで作成される倍長整数のメッセージ参照です。

blindCarbon は一つ以上のメールアドレスで構成されるAddressListです。

deleteOption は"Bcc"ヘッダに値を追加するか削除するかをしている整数値です。

- 0を指定すると、"Bcc" フィールドに新しい値を追加します。
- 1を指定すると、以前に設定されていた値を上書きして、"Bcc" フィールドに新しい値を設定します。(空の文字列*blindCarbon*に渡すと、ヘッダがメッセージから取り除かれます。)
- 2を指定すると、"Bcc" フィールドに設定されていた値が取り除かれ、ヘッダがメッセージから取り除かれます。

deleteOption はオプションの引数で、省略した場合のデフォルト値は0です。

例題

この例題では1つのメッセージを構築します。Forループの外側で静的なメッセージを部分を作成し、ループ内で[People]テーブルの各レコード毎に、メールアドレスをBCCに追加します。

```
$error:=SMTP_From($smtp_id;"sales@massmarket.com")
$error:=SMTP_Subject($smtp_id;"Terrific Sale! This week only!")
$error:=SMTP_Body($smtp_id;$GenericBody)
For($i;1;Records in selection([People]))
   $error:=SMTP_Bcc($smtp_id;[People]Email;0)  `このメールアドレスをBCCリストに追加
   NEXT RECORD([People])
End for
$error:=SMTP_Send($smtp_id)  `メッセージを送信
$error:=SMTP_Clear($smtp_id)
```

SMTP_Body

SMTP_Body (smtp_ID ; msgBody ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	➡	メッセージ参照
msgBody	テキスト	➡	メッセージ本文
deleteOption	整数	➡	0 = 置き換え (msgBodyが空でない場合), 1 = 削除, 2 = 追加
戻り値	整数	⮌	エラーコード

説明

SMTP_Body コマンドは、*smtp_ID*で指定されるメッセージのボディ部に、*msgBody*のテキストを設定します。

smtp_ID は*SMTP_New* コマンドで作成されるメッセージ参照です。

msgBody はメッセージのボディを含むテキスト値です。*msgBody*のサイズは32Kに制限されます。しかしこれはメールメッセージに32Kの制限があるということではありません。32K以上のボディを含むメッセージを送信するには、*deleteOption* 引数の追加フラグを使用します (下記参照)。実際の制限は利用可能なメモリによります。

警告： 通常、メッセージボディに (é, ö, etc.のような) アクセント文字を含めるべきではありません。これらの文字を使用する際は、*SMTP_SetPrefs* や *SMTP_Charset* コマンドの説明を参照してください。

deleteOption はボディを置き換えるか削除するか指定する整数値です:

- 0を指定すると、以前の値を置き換えて、ボディに新しい値を設定します。(空の文字列を*msgBody*に指定すると、以前のボディが保持されます。)
- 1を指定すると、以前の値を置き換えて、ボディに新しい値を設定します。(空の文字列を*msgBody*に指定すると、ボディが削除されます。)
- 2を指定すると、*SMTP_Body*によりボディに設定されていた値に、テキストが追加されます。

deleteOption はオプションの引数で、指定しない場合の値はデフォルトで0です。

例題

SMTPの完全な例題は以下のとおりです:

```
C_LONGINT($SMTP_ID)
C_BOOLEAN($SentOK;$OK)
$SentOK:=False  `すべてのコマンドが実行されたかを検証するフラグ
Case of
   :(Not(ERRCHECK("SMTP_New";SMTP_New($SMTP_ID))))
   :(Not(ERRCHECK("SMTP_Host";SMTP_Host($SMTP_ID;<>pref_Server))))
   :(Not(ERRCHECK("SMTP_From";SMTP_From($SMTP_ID;vFrom))))
   :(Not(ERRCHECK("SMTP_To";SMTP_To($SMTP_ID;vTo))))
   :(Not(ERRCHECK("SMTP_Cc";SMTP_Cc($SMTP_ID;vCC))))
   :(Not(ERRCHECK("SMTP_Bcc";SMTP_Bcc($SMTP_ID;vBcc))))
   :(Not(ERRCHECK("SMTP_Subject";SMTP_Subject($SMTP_ID;vSubject))))
   :(Not(ERRCHECK("SMTP_Comments";SMTP_Comments($SMTP_ID;"Sent via 4D"))))
   :(Not(ERRCHECK("SMTP_AddHeader";SMTP_AddHeader($SMTP_ID;"X-4Ddemo:";<>VERSION))))
   :(Not(ERRCHECK("SMTP_Body";SMTP_Body($SMTP_ID;vMessage))))
   :(Not(ERRCHECK("SMTP_Send";SMTP_Send($SMTP_ID))))
   Else
```

```
      $SentOK:=True  `メッセージが構築され、メールが送信された
End case

If($SMTP_ID#0)  `メッセージが作成されていたら、削除しなければなりません
   $OK:=ERRCHECK("SMTP_Clear";SMTP_Clear($SMTP_ID))
End if
```

Note: この **Case of** の特別な使用法については**Appendix A, プログラムTips**を参照してください。

以下は***ERRCHECK***メソッドのコードです。このメソッドは二つの引数をとります。第一引数はコマンド名 ($Command) で、第二引数は引数に渡されたSMTPコマンドが返すエラー値です。このメソッドからはエラー値が0の場合にFalseが、それ以外の場合はTrueが返されます。すなわちSMTPコマンドの実行に成功すれば、エラー値が0なので返り値がFalseとなり、次のCaseテストに進みます。SMTPコマンドの実行に失敗すると返り値がTrueとなるため、その時点でCaseのテストが終了します。

```
C_TEXT(vErrorMsg)
$Command:=$1
$Error:=$2
$Result:=True
If($Error#0)
   $Result:=False
   If(<>SHOWERRORS)  `エラーメッセージを表示するかしないか
      vErrorMsg:=IT_ErrorText($Error)
      ALERT("ERROR ---"+Char(13)+"Command: "+$Command+Char(13)+"Error
      Code"+String($Error)+Char(13)+"Description"+vErrorMsg)
   End if
End if
$0:=$Result
```

SMTP_Cc

SMTP_Cc (smtp_ID ; carbonCopy ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	➡	メッセージ参照
carbonCopy	テキスト	➡	MailAddress または AddressList
deleteOption	整数	➡	0 = 追加, 1 = 置き換え, 2 = 削除
戻り値	整数	⮌	エラーコード

説明

SMTP_Cc コマンドは、*smtp_ID*で指定されるメッセージにカーボンコピーの受信者を追加します。ccフィールドへのアドレスの追加は必須ではありません。"To" および "cc" ヘッダにリストされたすべてのアドレスは、メッセージの受信者に表示されます。

smtp_ID は*SMTP_New* コマンドで作成されるメッセージ参照です。

carbonCopy は一つ以上のメールアドレスからなるAddressListです。

deleteOption は"cc"ヘッダを追加するか削除するかを指定する整数値です。

- 0を渡すと、新しい値を"cc"フィールドに追加します。
- 1を渡すと、以前の値を上書きして、"cc"フィールドに新しい値を設定します。(空の文字列を carbonCopyに渡すと、このヘッダはメールから取り除かれます。)
- 2を渡すと、"cc"フィールドに設定されていた値が削除され、このヘッダはメールから取り除かれます。

deleteOption はオプションの引数で、指定しない場合のデフォルト値は0です。

例題

*SMTP_Body*コマンドの例題を参照してください。

SMTP_Charset

SMTP_Charset (encodeHeaders ; bodyCharset) -> 戻り値

引数	型		説明
encodeHeaders	整数	➡	-1 = 現在の設定を使用, 0 = 管理しない, 1 = ISO-8859-1またはISO-2022-JPの場合、指定された文字セットを使用して変換、拡張文字をエンコード
bodyCharset	整数	➡	-1 = 現在の設定を使用, 0 = 管理しない, 1= ISO-8859-1またはISO-2022-JPの場合、指定された文字セットを使用して変換
戻り値	整数	↩	エラーコード

説明

SMTP_Charset コマンドは、*SMTP_QuickSend* や *SMTP_Send* コマンドでメッセージを送信する際に、拡張文字を含むメッセージをサポートするために自動で変換処理が行われるよう設定します。このコマンドが呼び出されないか引数に0が渡されると、バージョン 6.7以降の4D Internet Commandsはバージョン6.5.xと同様に動作します。

SMTP_Charset コマンドは、まずメッセージヘッダとボディの変換に*SMTP_SetPrefs*の*bodyType* 引数を適用するかを指定し、次に拡張文字を含むヘッダをRFC#1342で定義された“=?ISO-8859-1?Q?Test=E9?= …”のシンタックスでエンコードするかを指定します。このコマンドはインタープロセススコープを持っていて、このコマンド実行後はすべての4Dプロセスで *SMTP_QuickSend* と *SMTP_Send*を使用して送信されるメッセージに影響します。

このコマンドは件名やメールアドレスに拡張文字が含まれる場合に有効です (たとえば“=?ISO-8859-1?Q?Test=E9?= <test@n.net >”のようなアドレスのエンコーディング)。

メッセージヘッダに基づきエンコーディング (Subjectヘッダを除き常にBase64に設定され、*SMTP_SetPrefs*の*bodyType* 引数値に基づく) は以下のように管理されます:

- Subject, Comment (“非構造化ヘッダ”): 拡張文字を含む場合、文字列全体がエンコードされる。
- From, To, CC, Bcc, Sender, ReplyTo, InReplyTo (“構造化ヘッダ”):
  - 山括弧 (“<”, “>”) に挟まれたテキストはメールアドレスとして認識され、エンコードされません。
  - SPC < > ( ) @ , ; : " / ? . = などの特別文字や区切り文字はエンコードされません。
  - 特別文字や区切り文字に挟まれた文字列は、拡張文字が含まれていればエンコードされます。
  - アドレスの例:
    someone@somewhereはエンコードされません
    Michele <michele@somewhere>, Micheleのみエンコードされます

*encodeHeaders*は、メッセージ送信時のヘッダの変換処理の方法とエンコーディングを指定する引数です。デフォルト値は0です。

- -1:　現在の設定を使用
- 0:　管理しない
- 1:
  - *SMTP_SetPrefs*の*bodyType* 引数が文字コードISO-8859-1またはISO-2022-JPに設定されている場合、ヘッダはその文字セットを使用して変換されます。
  - そうでない場合、指定されている文字セットにかかわらず、ヘッダに拡張文字が含まれていれば、以下のシンタックスで変換されます (RFC#1342参照) : “=?? Base64 Encoding?Test=E9?= …”
  - 例外: Subjectヘッダは必要に応じて、*SMTP_SetPrefs* コマンドの*bodyType* に指定されたエンコード方法でエンコードされます。

Note: “X_…”で始まる拡張ヘッダはUS ASCIIのみで記述しなければなりません。

*bodyCharset*は、メッセージ送信時にメッセージボディの変換に使用する文字セットを指定する引数です。デフォルト値は0です。

- -1: 現在の設定を使用

- 0: 管理しない
- 1: *SMTP_SetPrefs*の*bodyType* 引数が文字コードISO-8859-1またはISO-2022-JPに設定されている場合、メッセージボディはその文字セットを使用して変換されます。

例題

この例題では、件名とボディがISO-8859-1文字セットを使用して変換され、件名はRFC 1342シンタックスでエンコードされます:

```
SMTP_SetPrefs(1;1;0)
$err:=SMTP_Charset(1;1)
$err:=SMTP_QuickSend("mymail.com";"myaddress";"destination";"the Euro €";"the Euro symbol is €")
```

SMTP_Clear

SMTP_Clear (smtp_ID) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	⇒	メッセージ参照
		⇐	成功すれば0
戻り値	整数	⮌	エラーコード

説明

SMTP_Clear コマンドはメッセージ作成のために確保されていたメモリ上の空間を解放します。すべての *SMTP_New* の呼び出しは対になる *SMTP_Clear* がなければなりません。

smtp_ID は*SMTP_New* コマンドで作成された倍長整数のメッセージ参照です。SMTPメッセージのクリアに成功すると、*SMTP_Clear* コマンドは *smtp_ID* 変数に0を返します。

例題

*SMTP_Body*コマンドの例題を参照してください。

SMTP_Comments

SMTP_Comments (smtp_ID ; comments ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	➡	メッセージ参照
comments	テキスト	➡	コメントテキスト
deleteOption	整数	➡	0 = 置き換え (commentsが空でない場合), 1 = 置き換え, 2 = 削除
戻り値	整数	⬅	エラーコード

説明

SMTP_Comments コマンドは、メッセージボディに変更を加えずに、メッセージにコメントを追加する方法を提供します。コメントはメッセージのヘッダ部にのみ現れます。多くのメールアプリケーションはメッセージヘッダをユーザに表示しません。

smtp_ID は*SMTP_New* コマンドで作成されるメッセージ参照です。

comments は、メールヘッダに置く情報を含んだテキストです。

警告： テキストにラインフィード (ascii=10)を含んでいてはいけません。含まれていると、それはヘッダ部の終わりかつボディ部の始まりを意味します。これに続くヘッダはボディ部に押し出され、サーバやクライアントで正しく解釈されません。ヘッダについての詳細はRFC#822を参照してください。

deleteOption は"Comments"ヘッダを置き換えるか削除するか指定する整数値です:

- 0を指定すると、以前の値を置き換えて、"Comments" フィールドに新しい値を設定します。(空の文字列を*comments*に指定すると、以前のヘッダが保持されます。)
- 1を指定すると、以前の値を置き換えて、"Comments" フィールドに新しい値を設定します。(空の文字列を*comments*に指定すると、ヘッダが削除されます。)
- 2を指定すると、"Comments" フィールドに設定されていた値が削除され、ヘッダがメッセージから取り除かれます。

deleteOption はオプションの引数で、指定しない場合の値はデフォルトで0です。

例題

*SMTP_Body*コマンドの例題を参照してください。

SMTP_Date

SMTP_Date (smtp_ID ; msgDate ; msgTime ; timeZone ; offset ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	➡	メッセージ参照
msgDate	日付	➡	このメッセージの作成日
msgTime	時間	➡	このメッセージの作成時刻
timeZone	整数	➡	ロケーションコード
offset	整数	➡	timeZone引数に依存します
deleteOption	整数	➡	0 = 追加/置き換え, 1 = 削除
戻り値	整数	↩	エラーコード

説明

SMTP_Date コマンドはメールを作成した日付、時間、地域を受け取り、 smtp_ID で指定したメッセージのDateヘッダを構築します。コマンドに渡される日付は、メッセージを送信するマシンの現在日付と時刻であるべきです。日付は特別なフォーマットで記述されるため、メールを受信したサーバは記述された日付と時間、ゾーン、オフセットに基づき解釈することができ、結果送信者の日付と時刻をローカルの日付と時刻に変換できます。

Note: メールメッセージにDateヘッダが含まれない場合、SMTPサーバが日付と時刻を追加するでしょう。すべてのSMTPメールメッセージはクライアントアプリケーションまたはSMTPサーバ付加したDateヘッダを持っています。

smtp_ID は*SMTP_New* コマンドで作成された倍長整数のメッセージ参照です。

msgDate はこのメッセージが作成された4Dの日付です。

msgTime はこのメッセージが作成された時刻です。

timeZone は送信者のタイムゾーンを指定します。このフィールドには0から6の値が指定可能です。

- 0を指定した場合、直接offset 引数に国際標準時からの時差を指定します。
- 1を指定した場合、MacintoshのPRAMに基づく時差を自動で設定します。*timeZone* が1の場合、*offset* 引数は必要ありません。Macintoshのタイムゾーンは日付と時間システム環境設定で設定されます。メールの送信時刻がデータベースシステムで重要なファクタとなる場合、開発者はこのオプションの正確性について検討する必要があります。
- 2から5は米国の4つのタイムゾーンに対応します。それぞれの値に対するoffset はタイムゾーンが夏時間か (offset = 1) そうでないか (offset = 0) を指定します。
- 6を指定した場合、時間はミリタリータイムで指定されます。以下のミリタリータイムの表は*offset*を決定します。送信者のミリタリータイムコードに基づく対応するオフセット値 (-12 から 12) を使用してください。

offset - この引数に渡される値は *timeZone* 引数に設定されるコードにより異なります。先の説明および以下の説明を参照してください。

コード	タイムゾーン	offset引数
0	+/- offset from UT	+/- 時間で表記されるオフセット
1	+/- offset from UT	Offset引数を使用しない。MacのPRAM設定を使用
2	EST - EDT	(0 = EST, 1 = EDT)
3	CST - CDT	(0 = CST, 1 = CDT)
4	MST - MDT	(0 = MST, 1 = MDT)
5	PST - PDT	(0 = PST, 1 = PDT)
6	ミリタリータイム	以下の表を参照

offset値	ミリタリタイムコード
0	Z
-1 から -9	A から I
-10 から -12	K から M
1 から 12	N から Y

略号の説明	
UT	Universal Time
EST	Eastern Standard Time
EDT	Eastern Daylight Time
CST	Central Standard Time
CDT	Central Daylight Time
MST	Mountain Standard Time
MDT	Mountain Daylight Time
PST	Pacific Standard Time
PDT	Pacific Daylight Time

deleteOption - 0を渡した場合、与えられた引数を使用してDateヘッダが追加される、または事前に設定されていたDateヘッダを置き換えます。1を指定すると、Dateヘッダは取り除かれます。他の値は無視されます。*deleteOption* はオプションの引数であり、指定されない場合のデフォルト値は0です。

SMTP_Encrypted

SMTP_Encrypted (smtp_ID ; encrypted ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	⇒	メッセージ参照
encrypted	テキスト	⇒	暗号化方法
deleteOption	整数	⇒	0 = 置き換え (encryptedが空でない場合), 1 = 置き換え, 2 = 削除
戻り値	整数	⇐	エラーコード

説明

SMTP_Encrypted コマンドは、メッセージのボディに対して使用された暗号のタイプを受信者に通知するために使用します。4D Internet Commandsはメールメッセージ暗号・複号の機能を提供しません。メッセージの暗号化は開発者が行います。メッセージボディの暗号化が (*SMTP_Body*よりも前に) 行われた場合、コマンドを使用して暗号化の方法を通知すべきです。

smtp_ID は*SMTP_New* コマンドで作成されるメッセージ参照です。

encrypted は、メッセージボディの暗号化に使用された方法を指定するテキスト値です。受信側のメールアプリケーションは、メッセージボディの複合化に必要な情報をこのヘッダから取得します。フォーマットについてはRFC#822を参照してください。

警告： テキストにラインフィード (ascii=10)を含んでいてはいけません。含まれていると、それはヘッダ部の終わりかつボディ部の始まりを意味します。これに続くヘッダはボディ部に押し出され、サーバやクライアントで正しく解釈されません。ヘッダについての詳細はRFC#822を参照してください。

deleteOption は"Encrypted"ヘッダを置き換えるか削除するか指定する整数値です:

- 0を指定すると、以前の値を置き換えて、"Encrypted" フィールドに新しい値を設定します。(空の文字列をencryptedに指定すると、以前のヘッダが保持されます。)
- 1を指定すると、以前の値を置き換えて、"Encrypted" フィールドに新しい値を設定します。(空の文字列をencryptedに指定すると、ヘッダが削除されます。)
- 2を指定すると、"Encrypted" フィールドに設定されていた値が削除され、ヘッダがメッセージから取り除かれます。

deleteOption はオプションの引数で、指定しない場合の値はデフォルトで0です。

SMTP_From

SMTP_From (smtp_ID ; msgFrom ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	➡	メッセージ参照
msgFrom	テキスト	➡	MailAddress または AddressList
deleteOption	整数	➡	0 = 追加, 1 = 置き換え, 2 = 削除
戻り値	整数	↩	エラーコード

説明

SMTP_From コマンドは、メッセージの"From" フィールドにリストされるメールアドレスのリストを構築します。このフィールドのアドレスはメッセージ作成や認証に責任のある人々です。通常、"From"ヘッダにはメッセージを作成/送信した人の一つのアドレスが含まれます。ただし複数の人によりメッセージが作成されたという状況では、"From"にそれらの人のアドレスをリストすることもあります。

"From"ヘッダは必須です。"From"ヘッダにアドレスが指定されていれば、"Sender"ヘッダはオプションです。

smtp_ID は*SMTP_New* コマンドで作成される倍長整数のメールメッセージ参照です。

msgFrom 一つ以上のメールアドレスからなるAddressListです。Fromヘッダに記載されたアドレスは受信者に表示されます。

返信先について: *smtp_ID* で指定されるメッセージに"ReplyTo"ヘッダが定義されていない場合、メッセージの返信先には"From"ヘッダに記載されたそれぞれのアドレスが指定されます。

deleteOption は"From"ヘッダを追加または削除するかを指定する整数値です。

- 0を指定した場合、"From" フィールドに新しい値を追加します。
- 1を指定した場合、"From" フィールドに新しい値が設定され、以前の値は置き換えられます。(*msgFrom*に空の文字列を渡すと、ヘッダはメールエンベロープから取り除かれます。)
- 2を指定した場合、"From" フィールドから以前に追加した値を取り除き、メールエンベロープからヘッダを取り除きます。

deleteOption はオプションの引数で、指定しない場合のデフォルト値は0です。

例題

この例題では、３人の人がメッセージの作成に関わっています。このメッセージへの返信メールには、"From"ヘッダの３人のアドレスが設定されます。

```
$From:="prez@acme.com, vp@acme.com, cfo@acme.com"
$Error:=SMTP_From($smtp_id;$From;0)
$Error:=SMTP_Subject($smtp_id;"Company Policy Change";0)
$Error:=SMTP_To($smtp_id;<>AllEmployee;0)
```

SMTP_GetPrefs

SMTP_GetPrefs (lineFeeds ; bodyType ; lineLength) -> 戻り値

引数	型		説明
lineFeeds	整数	⇦	0 = 追加しない, 1 = LineFeedを追加する
bodyType	整数	⇦	Body-Content-Type
lineLength	倍長整数	⇦	行の最大長
戻り値	整数	⇨	エラーコード

説明

SMTP_GetPrefs コマンドはSMTP環境設定の現在の値を返します。*SMTP_SetPrefs* が設定を変更していない場合、このコマンドはデフォルト値を返します。引数の完全な説明は*SMTP_SetPrefs*を参照してください。

lineFeeds にはメッセージ本文の改行の処理方法が返されます。

bodyType にはBody-Content-Type設定の現在値が返されます。値の説明は *SMTP_SetPrefs* の *bodyType* の表を参照してください。

lineLength にはメッセージ本文の一行の最大文字数が返されます。

SMTP_Host

SMTP_Host (smtp_ID ; hostName ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	⇒	メッセージ参照
hostName	文字	⇒	ホスト名またはIPアドレス
deleteOption	整数	⇒	0 = 追加または置き換え, 1 = 削除
戻り値	整数	⇐	エラーコード

説明

SMTPコマンドで作成され送信されるメールは、特定のSMTPサーバに送られなければなりません。 4D Internet Commands は直接受信者にメールを届けることはしません。メールはこのコマンドで指定されたサーバに送られるのです。SMTPサーバはアドレスを解析し、メッセージ発送のスケジューリングを担当します。

smtp_ID は *SMTP_New* コマンドで作成されたメールメッセージ参照です。

hostName はメッセージの配送を担当するSMTPサーバのホスト名またはIPアドレスです。

deleteOption はオプションの引数で、現在のホスト設定を削除するかどうかを指定します。0を渡すと、*hostName*で指定した値がホストとして設定されます。1を指定すると、*smtp_ID*で指定されたメッセージからホスト定義を削除します。これはオプションの引数で、省略された場合のデフォルト値は0です。

例題

SMTP_Body と *SMTP_Send*の例を参照してください。

SMTP_InReplyTo

SMTP_InReplyTo (smtp_ID ; inReplyTo ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	➡	メッセージ参照
inReplyTo	テキスト	➡	In-Reply-To テキスト
deleteOption	整数	➡	0 = 置き替え (inReplyToが空でない場合), 1 = 置き換え, 2 = 削除
戻り値	整数	⮌	エラーコード

説明

SMTP_InReplyTo コマンドは、返信メッセージが関連する、返信元のメッセージを指定します。

smtp_ID は*SMTP_New* コマンドで作成されるメッセージ参照です。

inReplyTo は、このメッセージが関連する元のメッセージを指定するための文字列です。この文字列のフォーマットについては、RFC#822を参照してください。

警告： テキストにラインフィード (ascii=10)を含んでいてはいけません。含まれていると、それはヘッダ部の終わりかつボディ部の始まりを意味します。これに続くヘッダはボディ部に押し出され、サーバやクライアントで正しく解釈されません。ヘッダについての詳細はRFC#822を参照してください。

deleteOption は"ReplyTo"ヘッダを置き換えるか削除するか指定する整数値です:

- 0を指定すると、以前の値を置き換えて、"ReplyTo" フィールドに新しい値を設定します。(空の文字列をinReplyToに指定すると、以前のヘッダが保持されます。)
- 1を指定すると、以前の値を置き換えて、"ReplyTo" フィールドに新しい値を設定します。(空の文字列をinReplyToに指定すると、ヘッダが削除されます。)
- 2を指定すると、"ReplyTo" フィールドに設定されていた値が削除され、ヘッダがメッセージから取り除かれます。

deleteOption はオプションの引数で、指定しない場合の値はデフォルトで0です。

SMTP_Keywords

SMTP_Keywords (smtp_ID ; keywords ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	➡	メッセージ参照
keywords	テキスト	➡	キーワードリスト
deleteOption	整数	➡	0 = 置き換え (keywordsが空でない場合), 1 = 置き換え, 2 = 削除
戻り値	整数	⮌	エラーコード

説明

SMTP_Keywords コマンドは、*smtp_ID* で指定したメッセージの"Keywords"ヘッダにキーワードを挿入するために使用します。

smtp_ID は*SMTP_New* コマンドで作成されるメッセージ参照です。

keywords は、キーワードまたはキーワードリストを含むテキスト値です。フォーマットについてはRFC#822を参照してください。

警告： テキストにラインフィード (ascii=10)を含んでいてはいけません。含まれていると、それはヘッダ部の終わりかつボディ部の始まりを意味します。これに続くヘッダはボディ部に押し出され、サーバやクライアントで正しく解釈されません。ヘッダについての詳細はRFC#822を参照してください。

deleteOption は"Keywords"ヘッダを置き換えるか削除するか指定する整数値です:

- 0を指定すると、以前の値を置き換えて、"Keywords" フィールドに新しい値を設定します。(空の文字列をkeywordsに指定すると、以前のヘッダが保持されます。)
- 1を指定すると、以前の値を置き換えて、"Keywords" フィールドに新しい値を設定します。(空の文字列をkeywordsに指定すると、ヘッダが削除されます。)
- 2を指定すると、"Keywords" フィールドに設定されていた値が削除され、ヘッダがメッセージから取り除かれます。

deleteOption はオプションの引数で、指定しない場合の値はデフォルトで0です。

SMTP_New

SMTP_New (smtp_ID) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	⇦	作成された新しいメッセージへの参照
戻り値	整数	⇨	エラーコード

説明

SMTP_New コマンド は、*SMTP_QuickSend* でメール送信を行う場合を除いて、SMTPメールを構築する際にまず最初に呼ばれるコマンドです。*SMTP_New* はメモリ上に新規メッセージを作成し、その参照を *smtp_ID* 倍長整数変数に返します。この後に続くSMTPコマンドはこの *smtp_ID* 参照を使用して、メッセージのヘッダやボディを構築し、*SMTP_Send*でメッセージを送信します。

すべての *SMTP_New* 呼び出しは、*SMTP_Clear* コマンドの呼び出しと対になっていなければなりません。メッセージを送信した後、*SMTP_Clear* を呼び出して、メッセージ作成のために確保されていたメモリを解放します。

smtp_ID は倍長整数のメッセージ参照です。このIDは、このメールメッセージに対する参照として使用されます。一度に複数のメッセージを新規に作成できます。このIDにより、続くSMTPコマンドはどのメールに対して動作を適用するか決定します。

Examples

SMTP_Body や *SMTP_Send* コマンドの例題を参照してください。

SMTP_QuickSend

SMTP_QuickSend (hostName ; msgFrom ; msgTo ; subject ; message {; sessionParam}{; port}{; userName ; password}) -> 戻り値

引数	型		説明
hostName	文字	→	ホスト名またはIPアドレス
msgFrom	テキスト	→	送信元MailAddress または AddressList
msgTo	テキスト	→	送信先MailAddress または AddressList
subject	テキスト	→	件名
message	テキスト	→	メッセージ本文
sessionParam	倍長整数	→	1 = SSLを使用, 0または省略 = SSLを使用しない
port	倍長整数	→	使用するポート番号
userName	テキスト	→	認証に使用するユーザー名
password	テキスト	→	認証に使用するパスワード
戻り値	整数	←	エラーコード

説明

SMTP_QuickSend コマンドは、一つのコマンドでメールの構築と送信を可能にします。メッセージに対する詳細なコントロールが必要な場合は、*SMTP_New* コマンドをはじめとするSMTP関連のコマンドを使用します。

hostName はSMTPサーバのホスト名またはIPアドレスで、配送するメッセージを中継するサーバです。

msgFrom は、メールの送信人を表す一つ以上の完全なMailAddress または AddressListです。Fromヘッダに書かれたすべてのアドレスは メッセージの受信者から閲覧可能です。

msgTo は、一つ以上の完全なAddressListを含みます。msgTo ヘッダに記載されたAddressListにそれぞれメッセージが送信されます。それぞれの受信者は、他の受信者のアドレスを見ることができます。

subject にはメールの件名を渡します。

警告： 通常、メッセージの件名には (é, ö などの) アクセント文字を含めるべきではありません。これらの拡張文字を使用したい場合は、*SMTP_SetPrefs* と *SMTP_Charset* コマンド の説明を参照してください。

*message*はメールメッセージのボディテキストを渡します。過去の経緯からメッセージサイズは32 KBに制限されます。

オプションの*sessionParam*引数を使用すると、接続に使用するSSLプロトコルのアクティベーションモードを設定できます:

- 0を渡すかこの引数を省略すると、接続は標準の非保護モードで行われます。認証後にサーバーからSSL/TLSへの更新を要求された場合、スイッチは自動で行われます (explicitモードのSSL/TLS)。
- 1を渡すと、SMTPサーバーへの接続はSSLで行われます (同期モード)、
- 2を渡すと、接続は標準の非保護モードで行われます。SSL/TLSへの更新は行われません。

例題 1

以下はこのコマンドの使用例です:

```
$Host:="www.4d.com"
$ToAddress:="adupont@4d.fr"
$FromAddress:="jsmith@4d.com"
$Subject:="Sales Report"
$Message:="Can you send me the sales report for January 2000? Thanks."
$Error:=SMTP_QuickSend($Host;$FromAddress;$ToAddress;$Subject;$Message;1)
If($Error#0)
   ALERT("Error: SMTP_QuickSend"+Char(13)+IT_ErrorText($Error))
End If
```

例題 2

MS Exchange serverにセキュアモードでメールを送信する例題:

```
$ServerName:="exchange.4d.com"
$MsgTo:="adupont@gmail.com"
$MsgFrom:="a.user@4d.com"
$Subject:="Test message"
$Message:="This is a test for sending a message in secure mode. Please do not reply."
$Error:=SMTP_QuickSend($ServerName;$MsgFrom;$MsgTo;$Subject;$Message;0;587;"a.user";"@!password@!")
```

SMTP_References

SMTP_References (smtp_ID ; references ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	⇒	メッセージ参照
references	テキスト	⇒	Reference Text
deleteOption	整数	⇒	0 = Replace (if references not empty), 1 = Replace, 2 = Delete
戻り値	整数	⇐	エラーコード

説明

SMTP_References コマンドは、メッセージが参照する追加の関連を指定します。

smtp_ID は*SMTP_New* コマンドで作成されるメッセージ参照です。

references は、参照を含む文字列です。この文字列のフォーマットについては、RFC#822を参照してください。

警告： テキストにラインフィード (ascii=10)を含んでいてはいけません。含まれていると、それはヘッダ部の終わりかつボディ部の始まりを意味します。これに続くヘッダはボディ部に押し出され、サーバやクライアントで正しく解釈されません。ヘッダについての詳細はRFC#822を参照してください。

deleteOption は"References"ヘッダを置き換えるか削除するか指定する整数値です:

- 0を指定すると、以前の値を置き換えて、"References" フィールドに新しい値を設定します。(空の文字列をreferencesに指定すると、以前のヘッダが保持されます。)
- 1を指定すると、以前の値を置き換えて、"References" フィールドに新しい値を設定します。(空の文字列をreferencesに指定すると、ヘッダが削除されます。)
- 2を指定すると、"References" フィールドに設定されていた値が削除され、ヘッダがメッセージから取り除かれます。

deleteOption はオプションの引数で、指定しない場合の値はデフォルトで0です。

SMTP_ReplyTo

SMTP_ReplyTo (smtp_ID ; replyTo ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	→	メッセージ参照
replyTo	テキスト	→	MailAddress または AddressList
deleteOption	整数	→	0 = 既存のリストに追加, 1 = 古い値を新しい値で置き換え, 2 = 指定された値を取り除く
戻り値	整数	←	エラーコード

説明

SMTP_ReplyTo コマンドはメッセージへの返信を管理するアカウントを指定するために使用します。通常メールへの返信は"From"ヘッダに記載されたアドレスに対して行われます。"ReplyTo"ヘッダを設定すると、メッセージへの返信先のデフォルト値を変更できます。

データベース開発者にとって、*SMTP_ReplyTo* はとてもパワフルなツールであり、自動メールへの返信先をコントロールするために使用できます。ユーザは、返信を追跡するために作成されたアカウントなど、FromやSenderのアドレス以外に返信メールが送られるよう要望するかもしれません。

smtp_ID は*SMTP_New* コマンドで作成される倍長整数のメールメッセージ参照です。

replyTo は一つ以上のアドレスからなるAddressListです。このフィールドにリストされたアドレスは、受信者のメールソフトで、デフォルトのメール返信先として使用されます。

deleteOption は *replyTo*にリストされたメールアドレスをどのように処理するかを指定する整数値です。

- 0を指定した場合、このヘッダに事前に設定された値に新しい値を追加します。
- 1を指定すると、事前に設定された値が新しい値で置き換えられます。replyTo が文字列なら、事前に設定された値は取り除かれ、ヘッダも削除されます。
- 2を指定した場合、指定した値が事前に設定した値から取り除かれます。

deleteOption はオプションの引数であり、指定しない場合のデフォルト値は0です。

例題

この例題では、 3 人の管理者がメッセージを作成し、秘書が実際にメールを送信します。このメッセージへの返信は秘書と"personnel_dept"に送信され、管理者には送られません。

```
$From:="prez@acme.com, vp@acme.com, cfo@acme.com"
$Error:=SMTP_From($smtp_id;$From;0)
$Error:=SMTP_Sender($smtp_id;"secretary@acme.com";0)
$Error:=SMTP_ReplyTo($smtp_id;"secretary@acme.com, personnel_dept@acme.com";0)
$Error:=SMTP_Subject($smtp_id;"Company Policy Change";0)
$Error:=SMTP_To($smtp_id;◊AllEmployee;0)
```

SMTP_Send

SMTP_Send (smtp_ID {; sessionParam}) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	➡	メッセージ参照
sessionParam	倍長整数	➡	1 = SSLを使用, 0または省略 = SSLを使用しない
戻り値	整数	⮌	エラーコード

説明

SMTP_Send コマンドは *smtp_ID* で参照されるメッセージを送信します。しかしメモリからデータをクリアすることはありません。

smtp_ID は *SMTP_New* コマンドで作成された倍長整数のメッセージ参照です。

オプションの*sessionParam*引数を使用すると、接続にSSLプロトコルを使用することができます:

- 1を渡すと、SMTPサーバーへの接続はSSLで行われます (同期モード)、
- 0を渡すかこの引数を省略すると、接続は標準の非保護モードで行われます。

STARTTLS (explicit モード) の利用に関する注記

バージョン13.2以降、4D Internet CommandsはexplicitモードのSTARTTLS接続をサポートします。これはまず標準モードで接続を確立し、認証フェーズののちSSL/TLS接続にアップグレードできることを意味します。このメカニズムについては例題2も参照してください。

- 最初の接続はデフォルトポート (25) でない非SSL/TLSポートで開始されなければなりません。初期SMTP接続に使用するポートを指定するため、**SMTP_Send** の前に **IT_SetPort** コマンドを呼び出さなければなりません。MS Exchange serverへの接続にはポート番号587を使用します。
- 接続を認証させるために**SMTP_Auth**コマンドを呼び出さなければなりません。MS Exchange Serverと接続を行う場合、4DICはLOGIN認証モードのみをサポートします。このモードを渡すか、デフォルトモードのままにしてサーバーがサポートする最も安全なモードを利用します:

```
$error:=SMTP_Auth($smtp_id;"user.name";"password";2) // LOGINモードで認証
v$error:=SMTP_Auth($smtp_id;"user.name";"password") // サーバーによるLOGINモードの指定
```

例題 1

この例題では、メッセージが作成され、スタティックな要素がループの外側で定義されています。そののち、[People] テーブルのレコード毎にメッセージがカスタマイズされ、送信されます。

```
$error:=SMTP_New($smtp_id)
$error:=SMTP_Host($smtp_id;"wkrp.com")
$error:=SMTP_From($smtp_id;"herb_tarlick@wkrp.com")
$error:=SMTP_ReplyTo($smtp_id;"bigguy@wkrp.com")
$error:=SMTP_Subject($smtp_id;"Discounts on Ad Space!")
FIRST RECORD([People])
For($i;1;Records in selection([People]))
   If([People]Sales2Date>100000)
      $Body:=<>BigDiscText
   Else
      $Body:=<>SmlDiscText
   End if
   $Body:=Replace string($BoilerPlate;"<Salutation>";[People]Firstname)
```

```
    $error:=SMTP_To($smtp_id;[People]Email;1) `"To" ヘッダを新しい値で置き換える
    $error:=SMTP_Body($smtp_id;$Body)
    $error:=SMTP_Send($smtp_id)
    NEXT RECORD([People])
End for
$error:=SMTP_Clear($smtp_id)
```

例題 2

この例題ではMS Exchange server に STARTTLSを使用してテストメッセージを送信します:

```
$error:=SMTP_New($smtp_id)
$error:=SMTP_Host($smtp_id;"exchange.4d.com")
$error:=SMTP_From($smtp_id;"username@4d.com")
$error:=SMTP_ReplyTo($smtp_id;"username@4d.com")
$error:=SMTP_Subject($smtp_id;"Message test")
$error:=SMTP_Auth($smtp_id;"username";"!%@password")
$Body:="This is a test for messages sent through the Exchange, please do not reply"
$error:=IT_SetPort(2;587) //標準のSTMPモード, Exchange Serverではポート587
$error:=SMTP_To($smtp_id;"recipient@gmail.com")
$error:=SMTP_Body($smtp_id;$Body)
$error:=SMTP_Send($smtp_id;0) // アップグレードモード
ALERT(String($error));
```

SMTP_Sender

SMTP_Sender (smtp_ID ; msgSender ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	➡	メッセージ参照
msgSender	テキスト	➡	MailAddress (1つのみ)
deleteOption	整数	➡	0 = 追加, 1 = 置き換え, 2 = 削除
戻り値	整数	⮌	エラーコード

説明

SMTP_Sender コマンドはメッセージ送信者のe-mailアドレスを追加します。これはメッセージの実際の著者と送信者が異なる場合、または著者グループのうちだれが実際にメッセージを送信したかを示すために使用されます。このフィールドは"Sender" フィールドの内容が"From" フィールドと重複するような場合には不要です。

コンピュータプログラムがメールメッセージの作成者であり送信者でもある場合、Senderヘッダにはプログラムの動作を管理する実際の人のメールアカウントを設定すべきです。

smtp_ID は*SMTP_New* コマンドで作成された倍長整数のメールメッセージ参照です。

msgSender はメッセージのSenderフィールドに書かれる、一つのMailAddressを指定します。このヘッダには一つだけメールアドレスを指定します。

deleteOption はSenderヘッダの追加または削除を指定する整数値です。

- 0を指定した場合、Senderフィールドに新しい値を追加します。
- 1を指定した場合、Sender フィールドに新しい値が設定され、事前に設定された値は上書きされます。(*msgSender*に文字列を渡すと、ヘッダは取り除かれます。)
- 2を指定した場合、Senderフィールドに設定されていた値は取り除かれ、Sender フィールドも消去されます。

deleteOption はオプションの引数で、指定しない場合のデフォルト値は0です。

例題

この例題では、 3 人の管理職がメールを作成しています。メッセージの送信は秘書が行っています。返信メールには"From"ヘッダに記載されている 3 人のメールアドレスが設定されます。

```
$From:="prez@acme.com, vp@acme.com, cfo@acme.com"
$Error:=SMTP_From($smtp_id;$From;0)
$Error:=SMTP_Sender($smtp_id;"secretary@acme.com";0)
$Error:=SMTP_Subject($smtp_id;"Company Policy Change";0)
$Error:=SMTP_To($smtp_id;<>AllEmployee;0)
```

SMTP_SetPrefs

SMTP_SetPrefs (lineFeed ; bodyType ; lineLength) -> 戻り値

引数	型		説明
lineFeed	整数	➡	1 = [デフォルト] 追加, 0 = 追加しない, -1 = 変更しない
bodyType	整数	➡	Body-Content-Type (1 = [デフォルト] 自動検知, -1 = 変更しない)
lineLength	倍長整数	➡	一行の最大長 (0 = [デフォルト] 自動検知, -1 = 変更しない)
戻り値	整数	↩	エラーコード

説明

SMTP_SetPrefs コマンドは、SMTPコマンドで送信するメッセージの環境設定を行うために使用します。このコマンドはグローバルスコープを持っていて、コマンド実行後SMTPコマンドで作成されたすべてのSMTPメッセージに影響します。設定可能オプションは、*SMTP_QuickSend*や*SMTP_Send*を使用してSMTPサーバに送信されるメールメッセージのフォーマットに影響します。環境設定はインタープロセススコープを持っていて、すべての4Dプロセスに有効です。

SMTPサーバは改行と行送り (CR/LF) のペアを行の終了と認識します。これは改行を行の終端と認識するMacOSアプリケーションと異なる点です。

lineFeeds は整数値で、メールメッセージの本文の改行をどのように処理するか指示します。0を渡すとメッセージ本文は変更されません。開発者は独自の行送り追加処理を行うことができます。1 (デフォルト設定) を渡すと、すべての改行がCR/LFのペアに置き換えられます。-1は環境設定の現在の設定を保持します。どのオプションを選択すべきか決定できない場合、デフォルト値の 1 を選択すべきです。

bodyType では、送信するメッセージ本文の文字セットと (Body-Content-Type) 、メッセージ本文に適用するエンコーディング (Content-Transfer-Encoding) を、以下の表に従って指定します。たとえば“US-ASCII & 7 bit” (値 2) はメッセージ本文の文字セットとしてUS ASCIIが使用されており、4DICはそのメッセージを7 bitエンコーディングでエンコードすることを意味します。*SMTP_SetPrefs* コマンドは指定された文字セットを使用してメッセージ本文を変換しないことに注意してください。この変換は必要に応じてユーザが行います。文字セットの変換を強制させたい場合は *SMTP_Charset* コマンドの説明を参照してください。

変更しない限りデフォルトの値は1で、SMTPコマンドが自動で適当な設定をメッセージボディの内容から決定します。

-1	変更しない
0	Application & binary; エンコードなし
1	デフォルト; メッセージの内容に基づき "US-ASCII & 7bit" または "ISO-8859-1 & quotable-printable" のどちらかを選択
2	US-ASCII & 7bit
3	US-ASCII & quotable-printable
4	US-ASCII & base64
5	ISO-8859-1 & quotable-printable
6	ISO-8859-1 & base64
7	ISO-8859-1 & 8bit
8	ISO-8859-1 & binary
9	Reserved
10	ISO-2022-JP (Japanese) & 7 bit
11	ISO-2022-KR (Korean) & 7 bit
12	ISO-2022-CN (Traditional & Simplified Chinese) & 7 bit
13	HZ-GB-2312 (Simplified Chinese) & 7 bit
14	Shift-JIS (Japanese) & base64
15	UTF-8 & quoted-printable
16	UTF-8 & base64

lineLength は、メッセージ本文中の一行あたりの最大文字数を指定します。SMTPコマンドは最大行数に達する前の一番近い単語の切れ目に改行/行送りのペアを挿入します。どんな数値でも指定できますが、80文字以下にすることをお勧めします。-1を指定すると、現在の設定のまま変更されません。

lineLength 引数のデフォルト値は0です。この場合、SMTPコマンドは bodyType に基づき、RFCに定義された推奨値を使用します。*lineLength* 引数が0の場合、改行は以下の表に基づき行われます:

本文タイプ	**改行位置**
Base64	76
Quoted-Printable	76
その他…	改行なし

改行しないメッセージを送信すると問題が発生するシステムがあるため、改行の挿入は強く勧められます。またメッセージの配送経路途中で、メッセージフォーマットを処理できないコンピュータがあると、メッセージが拒否されることがあることも知っておいたほうがよいでしょう。

例題

以下のコードはUTF-8メッセージをquoted printableでエンコードして送信します:

```
$err:=SMTP_SetPrefs(-1;15;-1)
$err:=SMTP_QuickSend(...)
```

SMTP_Subject

SMTP_Subject (smtp_ID ; subject ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	⇒	メッセージ参照
subject	テキスト	⇒	メッセージ件名
deleteOption	整数	⇒	0 = 置き換え (subjectが空でない場合), 1 = 置き換え, 2 = 削除
戻り値	整数	⇐	エラーコード

説明

SMTP_Subject コマンドは、*smtp_ID*で参照されるメッセージに件名を追加します。すでに件名が*SMTP_Subject* コマンドにより追加されている場合、以前の件名が新しい件名で上書きされます。

smtp_ID は*SMTP_New* コマンドで作成されるメッセージ参照です。

subject は、メッセージ内容の概略を表すテキスト値です。

警告： 通常、メッセージ件名に (é, ö, etc.のような) アクセント文字を含めるべきではありません。これらの文字を使用する際は、*SMTP_SetPrefs* や *SMTP_Charset* コマンドの説明を参照してください。

警告： テキストにラインフィード (ascii=10)を含んでいてはいけません。含まれていると、それはヘッダ部の終わりかつボディ部の始まりを意味します。これに続くヘッダはボディ部に押し出され、サーバやクライアントで正しく解釈されません。ヘッダについての詳細はRFC#822を参照してください。

deleteOption は"Subject"ヘッダを置き換えるか削除するか指定する整数値です:

- 0を指定すると、以前の値を置き換えて、"Subject" フィールドに新しい値を設定します。(空の文字列を*subject*に指定すると、以前のヘッダが保持されます。)
- 1を指定すると、以前の値を置き換えて、"Subject" フィールドに新しい値を設定します。(空の文字列を*subject*に指定すると、ヘッダが削除されます。)
- 2を指定すると、"Subject" フィールドに設定されていた値が削除され、ヘッダがメッセージから取り除かれます。

deleteOption はオプションの引数で、指定しない場合の値はデフォルトで0です。

例題

*SMTP_Body*コマンドの例題を参照してください。

SMTP_To

SMTP_To (smtp_ID ; msgTo ; deleteOption) -> 戻り値

引数	型		説明
smtp_ID	倍長整数	➡	メッセージ参照
msgTo	テキスト	➡	MailAddress または AddressList
deleteOption	整数	➡	0 = 追加, 1 = 置き換え, 2 = 削除
戻り値	整数	⮌	エラーコード

説明

SMTP_To コマンドはメッセージの主たる受信者を指定するために使用します。"To" および "cc" ヘッダにリストされたアドレスはそれぞれのメッセージ受信者に表示されます。

smtp_ID は*SMTP_New* コマンドで作成される、倍長整数のメールメッセージ参照です。

msgTo は一つ以上のアドレスを含むAddressListです。

deleteOption は、"To"ヘッダに追加するかまたは削除するかを指定する整数値です。

- 0を渡すと、新しい値を"To" フィールドに追加します。
- 1を渡すと、以前に設定されていた値を上書きして、"To" フィールドに新しい値を設定します。(*msgTo*に空の文字列を渡すと、このヘッダは取り除かれます。)
- 2を設定すると"To" フィールドに事前に設定されていた値は削除され、このヘッダはメール殻取り除かれます。

deleteOption はオプションの引数で、制定しない場合のデフォルト値は0です。

例題

*SMTP_Body*の例題を参照してください。

IC ユーティリティ

ユーティリティコマンド - 概要

この節のコマンドは、他の4D Internet Commandsをサポートするさまざまなユーティリティを提供します。これらのコマンドを使用すると、開発者はユーザマシンの環境やソフトウェアのバージョン、コンピュータのIPアドレスや状態を取得できます。

また他のコマンドは、エラーコードを解釈したり、ファイルのエンコードやデコード、すべてのセクションの多くのコマンドに影響を与えるデフォルトのタイムアウト値設定などができます。

IT_Decode

IT_Decode (fileName ; decodedFile ; decodeMode) -> 戻り値

引数	型		説明
fileName	テキスト	⇒	エンコードされたファイルのLocalPath
decodedFile	テキスト	⇒	LocalPathファイル定義
		⇐	デコードされたファイルのパス
decodeMode	整数	⇒	1 = BinHex, 2 = Base64 (データフォークのみ), 3 = AppleSingle, 4 = AppleDouble, 5 = AppleSingle および Base64, 6 = AppleDouble および Base64, 7 = UUEncode, 8 = MacBinary
戻り値	整数	⇐	エラーコード

説明

IT_Decode コマンドは*decodeMode* で指定された方法でファイルをデコードします。指定されたファイルは変更されずデコードされたコピーが作成されます。

fileName にはデコードするファイルへのフルパス名を渡します。この引数に空の文字列を渡すと、ファイルを選択ダイアログが表示されます。

decodedFile には以下を渡すことができます:

- デコードされたファイルの格納場所と名前を指定するフルパス名。
- デコードされたファイルを格納するフォルダを指定するフルパス。ファイル名は元のファイル名が使用されます。
- 空の文字列。この場合、*IT_Decode* コマンドはデコード対象ファイルと同階層にデコード済みファイルを作成します。

指定されているかいないかにかかわらず、デコードされたドキュメントの格納先パス名がこの引数に返されます。

decodeMode はファイルに適用するデコード方法を指定します。デフォルト値は1でbinhexデコードが指定されます。指定可能な方法は以下のとおりです:

コード	スキーム
1	BinHex
2	Base64 (Data fork only)
3	AppleSingle
4	AppleDouble
5	AppleSingle and Base64
6	AppleDouble and Base64
7	UUEncode
8	MacBinary

デコードにAppleDouble (コード 4 & 6) を使用する場合、このコマンドはリソースフォークファイル"%filename"を探します。

IT_Encode

IT_Encode (fileName ; encodedFile ; encodedMode) -> 戻り値

引数	型		説明
fileName	テキスト	⇒	ファイルへのLocalPath
encodedFile	テキスト	⇒	LocalPathファイル定義
		⇐	エンコードされた結果ファイルのパス
encodedMode	整数	⇒	1 = BinHex, 2 = Base64 (データフォークのみ), 3 = AppleSingle, 4 = AppleDouble, 5 = AppleSingle および Base64, 6 = AppleDouble および Base64, 7 = UUEncode,8 = MacBinary
戻り値	整数	⇐	エラーコード

説明

IT_Encode コマンドは、*encodeMode*で指定された方法でファイルをエンコードします。指定されたファイルは変更されず、エンコードされたコピーが作成されます。エンコードされたファイルは元の名前にエンコード方法を示す拡張子が付加されたファイル名で作成されます。Binhexエンコードの場合、拡張子".hqx"が追加されます。 Base64エンコードの場合、拡張子".b64"が追加されます。AppleSingleエンコードの場合、拡張子".as"が追加されます。

fileName にはエンコードするファイルへのフルパス名を渡します。この引数に空の文字列を渡すと、ファイルを選択ダイアログが表示されます。

encodedFile には以下を渡すことができます:

- エンコードされたファイルの格納場所と名前を指定するフルパス名。
- エンコードされたファイルを格納するフォルダを指定するフルパス (ファイル名なし)。ファイル名は元のファイル名にエンコード方法を示す拡張子が付けられたものとなります。
- 空の文字列。この場合、*IT_Encode* コマンドはエンコード対象ファイルと同階層にエンコード済みファイルを作成します。

指定されているかいないかにかかわらず、エンコードされたドキュメントの格納先パス名がこの引数に返されます。指定されたディレクトリで名前の衝突の可能性があるので、エンコードしたファイルの格納先としては、引数として渡した値ではなく、常にこの返された値を参照すべきです。

encodeMode はファイルに適用するエンコード方法を指定します。デフォルト値は1でbinhexエンコードが指定されます。指定可能なエンコーディングは以下のとおりです:

コード	スキーム
1	BinHex
2	Base64 (データフォークのみ)
3	AppleSingle
4	AppleDouble
5	AppleSingle および Base64
6	AppleDouble および Base64
7	UUEncode
8	MacBinary

エンコードがAppleDouble (コード 4 & 6) を使用する場合、二つのファイル"%filename" と "filename"が作成されます。

IT_ErrorText

IT_ErrorText (error) -> 戻り値

引数	型		説明
error	整数	⇒	他のコマンドから返されたエラーコード
戻り値	文字	⇐	Text of the error

説明

IT_ErrorText コマンドは*error*整数値を受け取り、エラーの説明テキストを返します。このコマンドは他の多くの4D Internet Commandsと異なり、数値を返さないことに注意してください。

error はエラーコードを表す数値です。

例題

以下の***ErrorCheck***ルーチンの例題は、エラーの原因を説明する警告メッセージを表示します。

```
  `Method: ERRCHECK ("Command Name"; Error# ) -> True/False
C_TEXT(vErrorMsg)
$Command:=$1
$Error:=$2
$Result:=True
If($Error#0)
   $Result:=False
   vErrorMsg:=IT_ErrorText($Error)
   ALERT("ERROR -- "+Char(13)+"Command: "+$Command+Char(13)+"Error Code:"+String($Error)
   +Char(13)+"Description: "+vErrorMsg)
End if
$0:=$Result
```

IT_GetPort

IT_GetPort (protocol ; port) -> 戻り値

引数	型		説明
protocol	整数	⇨	1 = FTP; 2 = SMTP; 3 = POP3; 4 = IMAP
port	整数	⇦	ポート番号
戻り値	整数	⇩	エラーコード

説明

IT_GetPort コマンドは、指定した*protocol*で4D Internet Commandsが使用するポート番号を返します。

IT_GetProxy

IT_GetProxy (protocol ; proxyKind ; proxyHostName ; proxyPort ; proxyUserID) -> 戻り値

引数	型		説明
protocol	整数	→	1 = FTP; 2 = SMTP; 3 = POP3; 4 = IMAP
proxyKind	整数	←	0 = なし; 1 = SOCKS
proxyHostName	文字	←	SOCKS Proxyホストのホスト名またはIPアドレス
proxyPort	整数	←	接続に使用するProxyポート
proxyUserID	テキスト	←	SOCKSのユーザID
戻り値	整数	↩	エラーコード

説明

IT_GetProxy コマンドは、指定したプロトコルで4D Internet Commandsが使用する、ルーティングに関連する現在の設定を返します。*IT_SetProxy* で設定を変更しない限り、デフォルトのステータスが返されます。引数に関する詳細な説明は、*IT_SetProxy*を参照してください。

protocol には調査するプロトコルを指定します。1はFTPを、2はSMTPを、3はPOP3を、4はIMAPプロトコルを示します。

proxyKind にはSOCKS Proxyホストが使用されているかどうかを示す現在の設定が返されます。1が返された場合、指定されたプロトコルのすべてのリクエストは、SOCKSホストを経由して行われます。0が返された場合、SOCKSホストは経由しません。

proxyHostName にはSOCKS Proxyホストのホスト名またはIPアドレスが返されます。

proxyPort には、SOCKS Proxyホストとの通信に使用されるポート番号が返されます。

proxyUserID にはユーザIDが返されます。

IT_GetTimeOut

IT_GetTimeOut (timeout) -> 戻り値

引数	型		説明
timeout	整数	⇦	Timeout seconds
戻り値	整数	⮌	エラーコード

説明

IT_GetTimeOut コマンドは現在のタイムアウト値を返します。タイムアウトの影響を受けるコマンドは*IT_SetTimeOut*コマンドの説明でリストされています。

timeout には現在のタイムアウト秒数が返されます。

IT_MyTCPAddr

IT_MyTCPAddr (ip_Address ; subnet) -> 戻り値

引数	型		説明
ip_Address	文字	⇦	ユーザマシンのIPアドレス
subnet	文字	⇦	IP形式のサブネットマスク
戻り値	整数	⇨	エラーコード

説明

IT_MyTCPAddr コマンドは、コマンドが実行されているマシンのIPアドレスを返します。

ip_Address にはIPアドレスが返されます。

subnet にはIPアドレスのサブネットマスクが返されます。

IT_Platform

IT_Platform –> 戻り値

引数	型		説明
戻り値	整数		プラットフォームタイプ (1 = Mac OS, 2 = Windows)

説明

IT_Platform は、4D Internet Commandsコードが現在実行されているプラットフォームを示す整数値を返します。Mac OSの場合1が、Windowsの場合2が返されます。

例題

```
C_BOOLEAN(◊ITnative)
◊ITnative:=(IT_Platform=1)
```

IT_PPPConnect

IT_PPPConnect (pppProfil) -> 戻り値

引数	型		説明
pppProfil	文字	⇒	ダイアルアップ名 = Mac OSでは空文字, Windowsの場合必要
戻り値	整数	⇐	エラーコード

説明

IT_PPPConnect コマンドは、Mac OS上ではカレントのダイアルアップ接続を、Windows上では*pppProfil*引数で指定したダイアルアップ接続を開きます。このコマンドは関数として動作し、接続が開かれなかった場合エラーを返します。

このコマンドは、オンラインで動作する一連のインターネットコマンドを実行するたびに、実行する必要があります。完了したら、接続を閉じるために*IT_PPPDisconnect*を実行しなければなりません。

PPP (Point-to-Point Protocol) はシリアルインターフェースを使用して二つのコンピュータ間の通信を行う際に使用するプロトコルです。とくに電話線でサーバと接続するパーソナルコンピュータで使用されます。例えばあなたのISPがPPP接続を提供する場合、プロバイダのサーバがあなたのリクエストに応答しインターネットに転送、インターネットからのレスポンスをあなたに返します。本質的には、PPPはTCP/IPパケットをパッケージ化し、実際にインターネット上のサーバと通信するマシンに転送します。

通常PPPはデファクトスタンダードなSerial Line Internet Protocol (SLIP) で、同期および非同期の通信をサポートします。PPPでは一つのラインを他のユーザと共有でき、エラー検知機能を備えています。選択ができる限りPPPの使用が推奨されます。

IT_PPPDisconnect

IT_PPPDisconnect (pppProfil) -> 戻り値

引数	型		説明
pppProfil	文字	⇒	ダイアルアップ名 = Mac OSでは空文字, Windowsの場合オプション
戻り値	整数	⇐	エラーコード

説明

IT_PPPDisconnect コマンドは、*IT_PPPConnect*で開かれた接続を閉じます。

pppProfil には、閉じるダイアルアップ接続を指定するテキストを渡します。

Windowsでは、この引数は複数のPPP接続が同時に開かれている場合に有効です。この引数を使用すると、ユーザのネットワーク設定にかかわらず、正しい実行が期待できます。

Windowsの場合

- ひとつだけ接続が開かれていて、*pppProfil*が渡されないか空文字の場合、*IT_PPPDisconnect*は開かれている接続を閉じます。
- 複数の接続が開かれていて、*pppProfil*が渡されないか空文字の場合、*IT_PPPDisconnect*はエラーを返し、接続は閉じません。
- 有効な*pppProfil*が渡された場合、開かれている接続数にかかわらず、指定された接続が閉じられます。

Mac OSの場合

この引数は考慮されません。

IT_PPPStatus

IT_PPPStatus (pppProfil) -> 戻り値

引数	型		説明
pppProfil	文字	⇒	ダイアルアップ名 = Mac OSでは空文字, Windowsの場合オプション
戻り値	整数	⇐	1: 接続中; 0: 接続処理中; -1: エラー

説明

IT_PPPStatus コマンドを使用して、*IT_PPPConnect* コマンドを使用してまたは手動で開いた接続のステータスを取得できます。

pppProfil はチェックを行う開かれた接続を指定するテキストです。

Windowsでは、この引数はオプションですが、指定することで、ネットワーク設定にかかわらず正しく動作することが期待できます。

Windowsの場合

- 有効な*pppProfil*が渡された場合、指定した接続のステータスが返されます。
- *pppProfil*が省略されたか空文字の場合、*IT_PPPStatus*は以下を返します:
  - 複数の接続が開かれている場合、-1。
  - 接続が一つだけ開かれている場合、その接続のステータス。

Mac OSの場合

この引数は考慮されません。

*IT_PPPStatus*は接続のステータスを示す整数値を返します:

- 接続している場合、1。
- 接続処理中の場合、0。
- 接続失敗または接続されていない場合、-1。

例題

```
  ` GetMessagesメソッド （このメソッドはプロセス内で実行される）
If(mPPPConnect($vPPPProfil;120))
   $vErrCode:=IT_MacTCPInit
   If($vErrCode=0)
      $vErrCode:=POP3_Login...
      ...
   Else
      ALERT("Connection failed")
   End if
End if

  ` mPPPConnectメソッド
C_BOOLEAN($0)  `接続したらTrueが、失敗したらFalseが返される
C_TEXT($1)  `Mac OSでは空文字, Windowsでは接続名を入力
```

```
C_INTEGER($2) `タイムアウト秒数

If(IT_PPPStatus=1)
   $0:=True `すでに接続している
Else
   $vTimeoutLength:=$2
   $vTimeout:=False
   $vErr:=IT_PPPConnect($1)
   If($vErr=0)
      $vStart:=Current time
      Repeat
         DELAY PROCESS(Current process;30)
         $vStatus:=IT_PPPStatus($1)
         $vTimeout:=((Current time-$vStart)>$vTimeoutLength)
      Until(($vStatus=1)|$vTimeout) `接続したかタイムアウトした
      If(Not($vTimeout))
         $0:=True `接続した
      End if
   End if `… $Err = 0
End if
```

IT_SetPort

IT_SetPort (protocol ; port) -> 戻り値

引数	型		説明
protocol	整数	⇒	1 = FTP ; 2 = SMTP ; 3 = POP3 ; 4 = IMAP ; 12 = SMTP SSL ; 13 = POP3 SSL ; 14 = IMAP SSL
port	整数	⇒	ポート番号
戻り値	整数	⇐	エラーコード

説明

IT_SetPort コマンド実行後、*protocol*の通信はすべて*port*で指定したポート番号で行われます。

IT_SetProxy

IT_SetProxy (protocol ; proxyKind ; proxyHostName ; proxyPort ; proxyUserID) -> 戻り値

引数	型		説明
protocol	整数	→	1 = FTP; 2 = SMTP; 3 = POP3; 4 = IMAP
proxyKind	整数	→	0 = なし; 1 = SOCKS
proxyHostName	文字	→	SOCKS Proxyホストのホスト名またはIPアドレス
proxyPort	整数	→	接続に使用するProxyポート
proxyUserID	テキスト	→	SOCKSのユーザID
戻り値	整数	←	エラーコード

説明

IT_SetProxy コマンドを使用すると、SOCKSホスト (SOCKS Proxy) を通して、指定したプロトコルのリクエストを送信できるようになります。イントラネットの接続するだけの場合、SOCKSホストを通して通信する必要はないでしょう。しかしこれはすべて組織のファイウォールの設定に依存します。*IT_SetProxy*による設定はインタープロセススコープを持っていて、すべての4Dプロセスで、指定したプロトコルによる通信に影響します。

Note: SOCKSは、組織のネットワークでクライアントからのリクエストを受け入れ、インターネットに転送するためにProxyサーバが使用するプロトコルです。ワークステーションがファイアウォールの内側にあり、インターネット上の情報にアクセスしたい場合、SOCKSホストはリクエストを受信し、そのリクエストを転送して、情報をクライアントアプリケーションに返します。

protocol には、リクエストを指定したSOCKS Proxyホストを経由させるプロトコルを指定します。1はFTPを、2はSMTP、3はPOP3、4はIMAPプロトコルを示します。

proxyKind は、指定したプロトコルのリクエストがSOCKS Proxyホストを経由すべきかどうかを指定する整数値です。1を渡すと、指定したプロトコルのすべてのリクエストは、指定したSOCKSホストを経由します。0を渡すと指定したプロトコルのリクエストはSOCKSホストを経由しません。

proxyHostName はSOCKS Proxyホストのホスト名またはIPアドレスです。

proxyPort には、指定したプロトコルがSOCKS Proxyホストとの通信に使用するポート番号を渡します。

proxyUserID にはユーザを特定するテキスト値を渡します。ユーザIDはネットワーク管理者によって与えられます。*proxyUserID* が空の文字列の場合もあります。

例題

以下のメソッドでは、すべてのFTP接続は指定したSOCKS Proxyホストを経由します。

```
$err:=IT_SetProxy(1;1;$proxyAdd;$proxyPort;"") `FTP SOCKS Proxy
$err:=FTP_Login("ftp.4d.com";"anonymous";dbody@aol.com";$ftpID)
$err:=FTP_GetFileInfo($ftpID;$vpath;$vsize;$vmodDate)
$err:=FTP_Receive($ftpID;$vpath;"";0)
$err:=FTP_Logout($ftpID)
```

Note: 簡単にするために、この例題ではエラーチェックを行っていません。

以下の文はSOCKS Proxyホストを経由したFTP接続の転送を停止します。

```
$err:=IT_SetProxy(1;0;$proxyAdd;$proxyPort;"")
```

IT_SetTimeOut

IT_SetTimeOut (timeout) -> 戻り値

引数	型		説明
timeout	整数	→	タイムアウト秒数; 0から127まで
戻り値	整数	↩	エラーコード

説明

IT_SetTimeOut コマンドはタイムアウトを秒単位で設定します。この値は0から127の間でなければなりません。デフォルト値は30秒です。

timeout にはタイムアウト秒数を指定します。以下のコマンドが影響を受けます:

TCP_Open
FTP_Login
FTP_Send
FTP_Receive
SMTP_QuickSend
SMTP_Send
POP3_Login
POP3_BoxInfo
POP3_Delete
POP3_Reset
POP3_MsgInfo
POP3_MsgLstInfo
POP3_GetMessage
POP3_MsgLst
POP3_Download
POP3_VerifyID
POP3_UIDToNum
IMAP_Login
IMAP_VerifyID
IMAP_Capability
IMAP_ListMBs
IMAP_SubscribeMB
IMAP_GetMBStatus
IMAP_SetCurrentMB
IMAP_Delete
IMAP_MsgInfo
IMAP_MsgLstInfo
IMAP_GetMessage
IMAP_MsgLst
IMAP_SetFlags
IMAP_GetFlags
IMAP_Search
IMAP_MsgFetch
IMAP_Download
IMAP_CopyToMB
IMAP_CreateMB
IMAP_RenameMB
IMAP_DeleteMB
NET_Finger
NET_Ping

Note: *timeout*を0に設定すると、*TCP_Listen* コマンドは永久に接続を待ちうけます。このコマンドのためにタイムアウトを0に設定した後は、他の値に戻すことを忘れないようにしてください。また、タイムアウトは"TCP/IP タイムアウト" および "レスポンス待ち時間"の両方に影響します。タイムアウトを0に設定すると、レスポンスを待ちうけるのに十分な時間が与えられることはありません。

IT_TCPversion

IT_TCPversion (stackKind ; stackVersion) -> 戻り値

引数	型		説明
stackKind	整数	⇦	0 = なし, 3 = WinSock, 4 = BSD
stackVersion	テキスト	⇦	TCPスタックのバージョン番号
戻り値	整数	⇨	エラーコード

説明

IT_TCPversion コマンドは、4D Internet Commandsが現在使用しているTCPスタックのタイプに関する情報を返します。スタックのタイプはプラットフォームにより異なります。 Macintoshでは現在BSDのみがサポートされています。WindowsではWinSock TCPスタックがサポートされています。

stackKind には現在使用されているTCPスタックのタイプを示す整数値が返されます。サポートされる以下のTCPスタック値が返されます:

コード	TCPスタック
0	なし
1	MacTCP (無効, 互換性メモ参照)
2	Open Transport (無効, 互換性メモ参照)
3	WinSock
4	BSD Sockets

互換性メモ:

- MacTCPはもうサポートされていません (バージョン6.8より)。*stackKind* 引数に1が返されることはありません。
- Open Transportはもうサポートされていません (バージョン2004より)。*stackKind* 引数に2が返されることはありません。

stackVersion には現在使用され、*stackKind* 引数で示されるTCPスタックのバージョンを示す文字列が返されます。

IT_Version

IT_Version -> 戻り値

引数	型		説明
戻り値	文字		Version String

説明

IT_Version 関数は4D Internet Commandsのバージョンを示す文字列を返します。

例題

以下の例題では、4D Internet Commandsのバージョンを警告ダイアログに表示します。

```
ALERT("4D Internet Commands version: "+IT_Version)
```

IT_MacTCPInit

IT_MacTCPInit –> 戻り値

引数	型		説明
戻り値	整数		エラーコード

互換性に関する注意

IT_MacTCPInit は効果がなく、もう使用することはできません。

IT_MacTCPVer

IT_MacTCPVer (versionCode) -> 戻り値

引数	型		説明
versionCode	整数	⇦	インストールされているMacTCPのバージョンコード
戻り値	整数	⇨	エラーコード

互換性に関する注意:

*IT_MacTCPVer*コマンドは廃止されました。より互換性の高い*IT_TCPversion*を使用してください。

説明

4D Internet Commands バージョン 6.8よりMacTCPはサポートされていません。そのためプラットフォームやOSにかかわらず、*versionCode* 引数には0が返されます。

Appendix

Case文でコマンドを実行する

このドキュメントの多くの例題で、デベロッパがあまり目にしないようなプログラム構造が使用されています。つまりCase文で一連のコマンドを実行する方法です。

4D Internet Commandsの多くのコマンドは、完全に実行するために連続したコマンドの実行がすべて成功する必要があります。途中でコマンドの実行に失敗した場合、その後の実行は停止されなければなりません。If条件文でこれを記述するとカスケードが深くなってしまいます:

```
If(SMTP_New($smtp_id)=0)
   If(SMTP_Host($smtp_id;◊pref_Server)=0)
      If(SMTP_From($smtp_id;vFrom)=0)
         If(SMTP_To($smtp_id;vTo)=0)


         End if
      End if
   End if
End if
```

これに代わるものとして、4DのCase文を使用できます。Case文のそれぞれの条件式が、**True** または **False**を返すかどうか検証するために評価されます。すべてのCaseテストがFalseを返すと、すべてのテストが実行されます。先の例をCaseで書き直すと、以下のようになります:

```
$SentOK:=False  `すべての呼び出しを実行したかを示すフラグ
Case of
   :(SMTP_New($smtp_id)#0)
   :(SMTP_Host($smtp_id;◊pref_Server)#0)
   :(SMTP_From($smtp_id;vFrom)#0)
   :(SMTP_To($smtp_id;vTo)#0)
   :(SMTP_Subject($smtp_id;vSubject)#0)
   :(SMTP_Body($smtp_id;vMessage)#0)
   :(SMTP_Send($smtp_id)#0)
   Else
      $SentOK:=True  `メッセージが構築され送信された
End case
If($smtp_id#0)  `メッセージエンベロップが作成されていればクリアする
   $OK:=SMTP_Clear($smtp_id)
End if
```

上の例題では、すべての4D Internet Commandsのコマンドは、実行に成功すると0を返します。4Dはそれぞれのcase文を評価するために、4D Internet Commandsのコマンドを実行し、返り値を受け取ります。それぞれのcase文は返り値を0でないと比較しているため、コマンドの実行に失敗し非0値が返されると、case文の評価がTrueとなり、そこでテストが終わります。すべてのコマンドの実行に成功すると、4Dは**Else**句を実行し、$SentOKフラグがメッセージの送信に成功したことを示すように設定されます。

POP3 や IMAPメールでの自動応答メールに関する推奨

データベースに、受信したメールに対する返信が可能なメールシステムの実装を計画している場合、返信メッセージのフィー

ルドをどのように埋めるかということに対する標準的な推奨事項があります。以下の推奨はRFC#822で説明されています:

- "Sender" フィールドにリストされるアドレスはメッセージの配送中に発生した問題に関する通知を受け取ります。"Sender" フィールドが存在しない場合、通知は"From" フィールドにリストされたアドレスに送付されます。"Sender"メールアドレスにはメール配送に関する問題に関連する返信のみが送られるべきであり、メッセージのトピックに関連する返信は送られるべきではありません。
- "Sender"アドレスは自動的なメッセージ返信には使用されるべきではありません。かわりに"Reply-To" や "From" フィールドを以下の条件に基づき使用します。
- "Reply-To" フィールドが存在し、一つ以上のアドレスが含まれる場合、すべての返信はこのアドレスリストに送信します。"Reply-To"ヘッダのアドレスは"From"ヘッダにリストされるすべてのアドレスを上書きします。しかし"Reply-To" フィールドが存在せず、"From" フィールドが存在する場合、返信は"From"のアドレスに送信します。

この推奨は、返信を行う際のメールアドレスをプログラムでどのように決定するべきかという決定プロセスを助言するためのものです。返信メッセージが構築された後は、エンドユーザはメッセージ送信前にこのデフォルト返信先を上書きすることができます。

Appendix B, TCPポート番号

How to choose a port number

- 0 から 1023 (Well Known ポート): Well Known ポートはI.A.N.A. (Internet Assigned Numbers Authority) により割り当てられていて、ほとんどのシステムでsystemやrootプロセス、または権限を持つユーザによって実行されたプログラムのみが使用できます。
  - 20 と 21 FTP;
  - 23 TELNET;
  - 25 SMTP;
  - 37 NTP;
  - 80 と 8080 HTTP;
  - 443 HTTPS.
- 1024 から 49151 (登録済みポート): 登録済みポートはI.A.N.A.によりリストされ、ほとんどのシステムで通常のユーザプロセスや通常ユーザにより実行されたプログラム (ルータ, 特定のアプリケーション...) が使用できます。
- 49152 から 65535 (動的 / プライベートポート) : ダイナミック/プライベートポートは自由に使用できます。

データベースの同期をとるためにTCP/IPコマンドを使用する場合、49152以上のポート番号を使用しなければなりません。

詳細はI.A.N.A. Webサイトを参照してください: http://www.iana.org/

TCPポート番号

daytime	13	Daytime
qotd	17	Quote of the Day
ftp-data	20	File Transfer [Default Data]
ftp	21	File Transfer [Control]
telnet	23	Telnet
smtp	25	Simple Mail Transfer
time	37	Time
nicname	43	Who Is
domain	53	Domain Name Server
sql*net	66	Oracle SQL*NET
gopher	70	Gopher
finger	79	Finger
http	80	World Wide Web HTTP
poppassd	106	Password Server
rtelnet	107	Remote Telnet Service
pop2	109	Post Office Protocol - Version 2
pop3	110	Post Office Protocol - Version 3
sunrpc	111	SUN Remote Procedure Call
auth	113	Authentication Service
sftp	115	Simple File Transfer Protocol
sqlserv	118	SQL Services
nntp	119	Network News Transfer Protocol
ntp	123	Network Time Protocol
pwdgen	129	Password Generator Protocol
imap2	143	Interactive Mail Access Protocol v2
news	144	NewS
sql-net	150	SQL-NET
multiplex	171	Network Innovations Multiplex
cl/1	172	Network Innovations CL/1
at-rtmp	201	AppleTalk Routing Maintenance
at-nbp	202	AppleTalk Name Binding
at-3	203	AppleTalk Unused
at-echo	204	AppleTalk Echo
at-5	205	AppleTalk Unused
at-zis	206	AppleTalk Zone Information
at-7	207	AppleTalk Unused
at-8	208	AppleTalk Unused
ipx	213	IPX
netware-ip	396	Novell Netware over IP
timbuktu	407	Timbuktu
https	443	Secured protocol
conference	531	chat
netnews	532	readnews
netwall	533	for emergency broadcasts
uucp	540	uucpd
uucp-rlogin	541	uucp-rlogin
whoami	565	whoami
ipcserver	600	Sun IPC server
phonebook	767	phone

accessbuilder 888 AccessBuilder

Appendix C, 4D Internet Commandsエラーコード

すべての 4D Internet Commands (*IT_ErrorText* と *IT_Version*を除く) は実行結果として整数を返します。この整数値は4Dデータベースに伝える必要のあるエラー番号です。コマンドの実行に成功すると0が返されます。エラー発生元は通常エラー値の範囲により判断できます。以下の表で、一般的なエラーの発生元のインデックスを示します:

エラー番号	生成元
エラー < 0	OSエラーまたはWinSockネットワークレイヤ
0	エラーなし
エラー 1 -> 61	BSDネットワークレイヤ
エラー >= 10000	4D Internet Commands エラー

4D Internet Commands エラーコード

エラーが発生すると、以下の表で示すエラーコードが返されます:

10000	user cancelled a dialog or progress.
10001	unimplemented Internet command.
10002	invalid array type.
10003	no more (TCP,SMTP,POP3, etc.) references available.
10004	invalid reference.
10005	need a "Host" for use in the "SMTP_Send" command.
10006	need a "From" for use in the "SMTP_Send" command.
10007	need a recipient for use in the "SMTP_Send" command.
10008	already logged in.
10009	error trying to make a POP3 connection.
10010	error with POP3 USER.
10011	error with POP3 PASS.
10012	error with POP3 QUIT.
10013	error with POP3 STAT.
10014	error with POP3 LIST.
10015	error with POP3 UIDL.
10016	error with POP3 DELE.
10017	error with POP3 RSET.
10018	invalid message number.
10019	invalid character offset.
10020	invalid character length.
10021	error with POP3 RETR.
10022	field was not found in mail Header.
10023	no attachments found.
10024	error in processing BinHex.
10025	BinHex checksum error.
10026	Internet commands unavailable. Probably because MacTCP is not installed
10027	Connection no longer exists
10028	Exceeded 32k limit
10029	Error with POP3 NOOP
10030	POP3 session was closed by the server
10031	Error with POP3 APOP
10032	Unknown or invalid response.
10033	SMTP 421 - Service not available, closing transmission channel.
10034	SMTP 450 - Requested mail action not taken: mailbox unavailable.
10035	SMTP 451 - Requested action aborted: local error in processing.
10036	SMTP 452 - Requested action not taken: insufficient system storage.
10037	SMTP 500 - Syntax error, command unrecognized.
10038	SMTP 501 - Syntax error in parameters or arguments.
10039	SMTP 502 - Command not implemented.
10040	SMTP 503 - Bad sequence of commands.
10041	SMTP 504 - Command parameter not implemented.
10042	SMTP 550 - Requested action not taken: mailbox unavailable.
10043	SMTP 551 - User not local; please try <forward-path>.
10044	SMTP 552 - Requested mail action aborted: exceeded storage allocation.
10045	SMTP 553 - Requested action not taken: mailbox name not allowed.
10046	SMTP 554 - Transaction failed.
10047	FTP 421 - Service not available, closing control connection.
10048	FTP 425 - Can't open data connection.

10049	FTP 426 - Connection closed; transfer aborted.
10050	FTP 450 - Requested file action not taken. File unavailable (e.g.,file busy).
10051	FTP 451 - Requested action aborted: local error in processing.
10052	FTP 452 - Requested action not taken. Insufficient storage space in system.
10053	FTP 500 - Syntax error, command unrecognized.
10054	FTP 501 - Syntax error in parameters or arguments.
10055	FTP 502 - Command not implemented.
10056	FTP 503 - Bad sequence of commands.
10057	FTP 504 - Command not implemented for that parameter.
10058	FTP 530 - Not logged in.
10059	FTP 532 - Need account for storing files.
10060	FTP 550 - Requested action not taken. File unavailable (e.g., file not found, no access).
10061	FTP 551 - Requested action aborted: page type unknown.
10062	FTP 552 - Requested file action aborted. Exceeded storage allocation (for current directory or dataset).
10063	FTP 553 - Requested action not taken. File name not allowed.
10064	No response has been received within the given timeout period.
10065	Not an FTP file.
10066	Error in processing Base64.
10067	Error in processing AppleSingle.
10068	Error in processing Quoted-Printable.
10069	FTP session was closed by the server.
10070	Not an FTP directory.
10071	TCP session was closed by the server
10072	Invalid encode kind
10073	Invalid decode kind
10074	An asynchronous DNR call did not complete
10075	An asynchronous OpenTransport call did not complete
10076	OpenTransport bind failed
10077	OpenTransport connect failed
10078	Maximum MacTCP streams reached
10079	Error in processing uuencode
10080	Cannot load ICMP library
10081	Error in processing MacBinary
10082	MacBinary checksum error
10083	Could not open a file
10084	No FTP information received
10085	Unknown FTP information received
10086	Proxy connection failed
10087	Standard file I/O error
10088	FTP reentrant error
10089	SLI.DLL is not loaded
10091	Error trying to make an IMAP connection
10092	A maibox is not selected
10093	Invalid message part
10094	Error with IMAP LOGIN
10095	Error with IMAP LOGOUT
10096	Error with IMAP CAPABILITY
10097	Error with IMAP SELECT
10098	Error with IMAP FETCH

10099	Error with IMAP PARTIAL
10100	Error with IMAP STORE
10101	Error with IMAP EXPUNGE
10102	Error with IMAP SEARCH
10103	Error with IMAP COPY
10104	Error with IMAP CREATE
10105	Error with IMAP DELETE
10106	Error with IMAP RENAME
10107	Error with IMAP SUBSCRIBE
10108	Error with IMAP UNSUBSCRIBE
10109	Error with IMAP LIST
10110	Error with IMAP LSUB
10111	Error with IMAP STATUS
10112	Error with IMAP CLOSE
10113	Error with AUTHENTICATION

BSD Error Codes

1	Operation not permitted
4	Interrupted system call
13	Permission denied
14	Bad address
22	Invalid argument
24	Too many open files
35	Operation would block
36	Operation now in progress
37	Operation already in progress
38	Socket operation on non-socket
39	Destination address required
40	Message too long
41	Protocol wrong type for socket
42	Protocol not available
43	Protocol not supported
44	Socket type not supported
45	Operation not supported
46	Protocol family not supported
47	Address family not supported by protocol family
48	Address already in use
49	Can't assign requested address
50	Network is down
51	Network is unreachable
52	Network dropped connection on reset
53	Software caused connection abort
54	Connection reset by peer
55	No buffer space available
56	Socket is already connected
57	Socket is not connected
58	Can't send after socket shutdown
60	Operation timed out
61	Connection refused

WinSock Error Codes

-10004	Blocking call cancelled
-10013	Permission denied
-10014	Bad address
-10022	Invalid argument
-10024	No more sockets available
-10035	Non-blocking socket would block
-10036	Illegal WinSock function invoked while a blocking function is in progress
-10037	An attempt was made to cancel an asynchronous operation that has already completed
-10038	Specified socket descriptor is not valid for this application
-10039	Destination address was required but none was supplied to the function
-10040	Datagram too large for buffer
-10041	Specified protocol does not match the other parameters in the call
-10042	Protocol option is unknown or invalid
-10043	Specified protocol is not supported by the Windows Sockets implementation
-10044	Specified socket type is not supported by the specified address family
-10045	Socket does not support the specified operation
-10046	Protocol family not supported
-10047	Specified address family is not supported by the Windows Sockets implementation or cannot be used with the indicated socket
-10048	Specified address is already in use
-10049	Specified address is not available from the local machine
-10050	Problem with the network subsystem
-10051	Network cannot be reached from this host at this time
-10052	Connection was dropped and must be reset
-10053	Connection was aborted because of a timeout or other error condition
-10054	Connection was reset by the remote host
-10055	Windows Sockets implementation is out of buffer space or the space provided in an API call by the application was too small to hold the requested information
-	

-10056	Specified socket is already connected
-10057	Specified socket is not connected
-10058	Socket has had the requested functionality shut down
-10060	Connection attempt timed out before the connection could be established
-10061	Connection attempt was forcefully rejected
-10091	Network subsystem is not yet ready for communication
-10092	Windows Sockets DLL does not support the requested Winsock protocol version
-10093	Windows Sockets not initialized
-11001	Requested database information does not exist; as confirmed by an authoritative host
-11002	Requested information was not found but the answer was not authoritative
-11003	Non-recoverable error occurred
-11004	Name supplied was valid but no information of the requested type is in the database

SMTP RFC Values

The following items are **not** error codes returned by any of the external commands. These are response codes which the SMTP protocol has defined to communicate various states during client-server communication. Developers may find this list useful if they are writing their own mail communication procedures using low-level TCP commands.

211	System status, or system help reply
214	Help message [Information on how to use the receiver or the meaning of a particular non-standard command; this reply is useful only to the human user]
220	<domain> Service ready
221	<domain> Service closing transmission channel
250	Requested mail action okay, completed
251	User not local; will forward to <forward-path>
354	Start mail input; end with <CRLF>.<CRLF>
421	<domain> Service not available, closing transmission channel [This may be a reply to any command if the service knows it must shut down]
450	Requested mail action not taken: mailbox unavailable [e.g., mailbox busy]
451	Requested action aborted: local error in processing
452	Requested action not taken: insufficient system storage
500	Syntax error, command unrecognized [This may include errors such as command line too long]
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command parameter not implemented
550	Requested action not taken: mailbox unavailable [e.g., mailbox not found, no access]
551	User not local; please try <forward-path>
552	Requested mail action aborted: exceeded storage allocation
553	Requested action not taken: mailbox name not allowed [e.g., mailbox syntax incorrect]
554	Transaction failed

FTP RFC Values

The following items are **not** error codes returned by any of the external commands. These are response codes which the FTP protocol has defined to communicate various states during client-server communication. Developers may find this list useful when writing their own file transfer procedures using low-level TCP commands.

110	Restart marker reply. In this case, the text is exact and not left to the particular implementation; it must read: MARK yyyy = mmmm. Where yyyy is User-process data stream marker, and mmmm server's equivalent marker (note the spaces between markers and "=").
120	Service ready in nnn minutes.
125	Data connection already open; transfer starting.
150	File status okay; about to open data connection.
200	Command okay.
202	Command not implemented, superfluous at this site.
211	System status, or system help reply.
212	Directory status.
213	File status.
214	Help message on how to use the server or the meaning of a particular non-standard command. This reply is useful only to the human user.
215	NAME system type. Where NAME is an official system name from the list in the Assigned Numbers document.
220	Service ready for new user.
221	Service closing control connection. Logged out if appropriate.
225	Data connection open; no transfer in progress.
226	Closing data connection. Requested file action successful (e.g., file transfer or file abort).
227	Entering Passive Mode (h1,h2,h3,h4,p1,p2).
230	User logged in, proceed.
250	Requested file action okay, completed.
257	"PATHNAME" created.
331	User name okay, need password.
332	Need account for login.
350	Requested file action pending further information.
421	Service not available, closing control connection. This may be a reply to any command if the service knows it must shut down.
425	Can't open data connection.
426	Connection closed; transfer aborted.
450	Requested file action not taken. File unavailable (file busy).
451	Requested action aborted: local error in processing.
452	Requested action not taken. Insufficient storage space in system.
500	Syntax error, command unrecognized. This may include errors such as command line too long.
501	Syntax error in parameters or arguments.
502	Command not implemented.
503	Bad sequence of commands.
504	Command not implemented for that parameter.
530	Not logged in.
532	Need account for storing files.
550	Requested action not taken. File unavailable (e.g., file not found, no access).
551	Requested action aborted: page type unknown.
552	Requested file action aborted. Exceeded storage allocation (for current directory or dataset).
553	Requested action not taken. File name not allowed.

ここではインターネットプロトコルに関する情報が必要な際に参照できるWebサイトを紹介します。Webドキュメントには Webブラウザでアクセスできます。

http://www.internic.net/: ドメイン名について理解し、それを取得する方法について理解できます。

http://www.ietf.org/: Internet Engineering Task Force (IETF) サイト。

http://www.rfc-editor.org/: RFCについて理解し、RFCや関連するサイトを検索できます (http://www.rfc-editor.org/rfc.html)。

ftp://ftp.isi.edu/in-notes/rfc821.txt: Simple Mail Transfer Protocol -- RFC 821.

http://www.w3c.org/: World Wide Webについて知りたい場合に参照します。

http://www.imap.org: IMAPプロトコルのために予約されたサイト。このプロトコルに関する有用な情報を検索できます。

4D Internet Commands - 新着

13.2 | 13.0

- SMTP_QuickSend Updated 13.2
- SMTP_Send Updated 13.2

4D Internet Commands - コマンドリスト (文字順)

F | I | M | N | P | S | T | U | 引

- FTP_Append
- FTP_ChangeDir
- FTP_Delete
- FTP_GetDirList
- FTP_GetFileInfo
- FTP_GetPassive
- FTP_GetType
- FTP_Login
- FTP_Logout
- FTP_MacBinary
- FTP_MakeDir
- FTP_PrintDir
- FTP_Progress
- FTP_Receive
- FTP_RemoveDir
- FTP_Rename
- FTP_Send
- FTP_SetPassive
- FTP_SetType
- FTP_System
- FTP_VerifyID