4D Internet Commands

Reference Guide
Windows® and Mac™OS Versions

4D Internet Commands

Version 6.5 for Windows® and Mac™ OS

Contents

1 4D Internet Commands

Preface

4D Internet Commands version 6.5

4D Internet Commands empower users of 4th Dimension with a robust set of communication utilities capable of working in either a Local or Wide area network. The ultimate expression of this explosion of connectivity is known colloquially as "The Internet". The last few years has produced phenomenal growth in the number of people and companies gaining access to the internet. As the volume of people with internet access increases, the need to be "on the net" is felt stronger each day by those in the business community.

The suite of commands provided by 4D Internet Commands gives database developers access to many key elements of the Internet. The SMTP commands contain tools to automate e-mail delivery from a database to any list of people. Similarly, the POP3 commands can retrieve mail from any number of mailboxes for storage within the database, re-routing, auto-reply or remote-search execution. The FTP commands enable the user to transfer files to/from remote systems or to obtain directory listings of files on FTP volumes. And the TCP commands provide developers with the low-level tools enabling them to accomplish any internet-related task.

SMTP (Simple Mail Transfer Protocol) is the primary mail transfer protocol used over the internet. 4D Internet Tool allows users to quickly build and send mail via a SMTP server. Mail creation and delivery can be as simple as a single command. If your mail delivery needs are more complex, every aspect of the message header, body and attachments can be controlled to affect its delivery. Since internet mail addressing provides for delivery to such services as CompuServe, America Online, eWorld, etc. you are able to reach virtually anyone with an e-mail account. Other examples of how the suite of SMTP commands could be used are:

- Automation of database report delivery
- Creation of an automatic mail forwarding database
- Group mail-list management
- Store-and-forward remote database updates & synchronizations

Along with its SMTP commands, 4D Internet Commands also contains commands which will connect to POP3 (Post Office Protocol, Version 3) mail servers for retrieval of mail messages and encoded attachments. Since the suite of SMTP and POP3 commands conform to the MIME standard for multiple message enclosures, binary attachments can easily be downloaded and saved.

The commands also provide users with the ability to encode attachments in several different ways such as: Binhex, Base64, AppleSingle, AppleDouble...
The FTP (File Transfer Protocol) commands provide a very easy-to-use mechanism for communicating with an FTP server to send/receive text or binary files. Commands within the FTP suite can obtain directory listings of files, enabling the database developer to create navigable interfaces to remote volumes of files. The FTP commands can easily be used in document-tracking applications without requiring client applications to mount remote volumes directly.

Transmission Control Protocol/Internet Protocol, (TCP/IP) is the primary protocol used for sending and receiving data over the internet. 4D Internet Commands contains several commands for sending and receiving raw TCP packets. The TCP set of commands provide developers with the essential tools to build and control their own internet communications.

Some examples are:

- Turn your database into a WWW server
- Build your own telnet interface
- Execute shell commands on remote machines
- Retrieve documents from the World Wide Web
- Search through numerous on-line databases
- Handle database synchronizations with remote servers
- Federal Express and UPS package tracking

Installation

The “4D Internet Commands” plug-in is integrated in 4th Dimension in the same way plug-ins do.

4D Internet Commands becomes automatically available when you install an ACI 6.5 product since this plug-in is automatically installed at two different locations:
• in the current System folder (Windows\ACI\WIN4DX on Windows or System folder:Preferences:ACI:MAC4DX on Mac OS), which enables its immediate use with no further handling.
• in the standard installation folder, which allows users to easily move the plug-in or duplicate it independently of the product installation process.

Using this architecture, users can choose to install the 4D Internet Commands plug-in:
• by placing it in a Mac4DX or Win4DX folder at the same level as the database structure or,
• by leaving it in the current active System folder .

Important Installation Notes:
• If 4D Internet Commands is installed in both locations, only the version located in the WIN4DX/MAC4DX folder at the same level as the database structure will be loaded.
• Compiling: When compiling a database that uses 4D Internet Commands installed in the System folder, it is required that you specify the path to the 4D Internet Commands plug-in to 4D Compiler.
• When copying or moving a database or an executable on another machine, make sure that the WIN4DX/MAC4DX folder in which the plug-in is located is copied at the same time.

People who need to install 4D Internet Commands independently of an ACI 6.5 product will find the plug-in files at the root level of the ACI Product Line 6.5 CD Rom.

For more information, please refer to the ACI software *Installation Guide*.

Software Requirements

The list below details the software requirements to use 4D Internet Commands in a 4th Dimension database. It is not necessary to meet all of these requirements, only those pertaining to the set of commands which will be used. Virtually all of the external calls communicate via the TCP/IP protocol and a TCP/IP protocol stack should be considered a requirement across all functionality sets.

Hardware
- 680x0-based or PowerPC native
- Intel machine running Windows95/98 or WindowsNT

4th Dimension (Mac & Windows)
- 4D for Mac version 6.0 or greater
- 4D for Windows version 6.0 or greater

MacTCP, Open Transport, Winsock
All the commands throughout 4D Internet Commands use the TCP/IP protocol for communications. Any computer which is to execute commands within this plug-in must have a TCP/IP driver installed and properly configured with a unique IP address. Most operating systems have TCP/IP drivers pre-installed, example of which are MacTCP or Open Transport on the Macintosh and Winsock on Windows.
For more information on configuring TCP/IP, please refer to your network administrator or the *Network Components for 4D Server* Reference manual.

Note: Starting from version 6.5, 4D Macintosh Network Components only work with Open Transport ("Classic" Networking is no longer supported). However, 4D Internet Commands do not use the 4D TCP/IP Network Component to communicate with the TCP/IP protocol. So, 4D Internet Commands may use MacTCP driver. On another hand, if you want to use a 4D Client 6.5 or higher, Web features of 4D 6.5 or higher, you need to use the Open Transport TCP/IP driver on MacOS.

Network access
In order to use the suite of commands in 4D Internet Commands, you must have access to a network which supports TCP/IP.

Domain Name Server
For many of the 4D Internet Commands, it is necessary to have access to a domain name server. For more information on configuring TCP/IP, please refer to the *Network Components for 4D Server* Reference manual, or the Apple guide.

SMTP Mail Server
In order to send mail using the set of SMTP commands, it is necessary for the sender to have access to a SMTP mail server which will forward the message to a POP3 mail server.

POP3 Mail Server
In order to use the POP3 commands, you must have an account on a POP3 mail server.

Glossary and Terminology

4D Internet Commands

version 6.5

This section defines many of the references made throughout the manual. The definitions are simplistic and are meant mainly for those unfamiliar with the references. The Terminology section pertaining to "Parameter Formats" provides details on the formatting expectations of 4D Internet Commands common parameters.

NIC: "Network Information Center". For the most part, the internet is an unregulated entity. There is no centralized authority or control over its use or growth. However, there are some basic administrative needs such as domain name and IP address assignments which could only be effectively carried out if controlled by a single agency. The NIC is the group responsible for such administrative tasks.

RFC: "Request for Comments." Most of the 4D Internet Commands are based upon standards defined to handle internet communication. The standard methodologies, descriptions and protocols used throughout the internet are defined within documents known as RFCs. Appendix E, Additional Information contains references to some WWW sites with pointers to many of the RFC documents. The 4D Internet Commands package does its best to protect you from a need to reference these documents, though anyone programming their own communications via the low-level TCP routines should become familiar with them.

TCP/IP Addresses, Host Names and Domain Names: An IP address is a reference to a **specific** machine somewhere out in the world. The IP address is formatted as a string containing four numeric values separated by periods (i.e. "207.94.15.3"). Each numeric part of the address can contain a value between zero and 255. By applying some mathematical functions to an IP address, its value can be squeezed down into an equivalent Long Integer number which this document will refer to as the **ip_LongInt**.

In order for a site (i.e. a Company, College, etc.) to put their computers on the internet, some assurances must be taken that their IP addresses won't conflict with other machines on the network. Institutions (and often individuals) will register their site with the **NIC** in order to obtain a **Domain Name**. **Domain Names** provide a system of easy-to-remember Internet addresses, which can be translated by the Domain Name System (DNS) into the numeric addresses (Internet Protocol [IP] numbers) used by the network. This system allows a more readable format such as "www.acius.com" or "ftp.acius.com".

Domain Name = "acius.com"

Host Name (Name of a computer)	=	**IP address**	=	**ip_LongInt**
"www.acius.com"	=	"207.94.15.3"	=	-815919357

The relationship between a **Host name** and its corresponding IP address is stored in a database known as a DNS (Domain Name System). These servers communicate with one another to exchange any new or changed data in the domain name lists throughout the world. The TCP/IP control panel provides a means to 'point' your computer to a DNS, which will then resolve all domain name references you use.

It is important to understand that all domain name servers have a corresponding IP address. However, not all IP addresses have a corresponding domain name server. Also, a "Mail Address" such as "jsmith@acius.com" does not reference that person's specific computer or IP address. The mail address would direct its delivery to the machine with the IP address represented by resolving the domain "acius.com". The mail would be delivered to the POP3 server running on that machine, which would then hold the mail for its user named "jsmith".

Domain Name: The Domain Name is an addressing construct used for identifying and locating computers on the Internet. Domain names provide a system of easy-to-remember Internet addresses, which can be translated by the Domain Name System (DNS) into the numeric addresses (Internet Protocol [IP] numbers) used by the network. A domain name is hierarchical and often conveys information about the type of entity using the domain name. A domain name is simply a label that represents a domain, which is a subset of the total domain name space. Domain names at the same level of the hierarchy must be unique, for example there can be only one com at the top level of the hierarchy, and only one acius.com at the next level of the hierarchy. If your organization's name is "CompanyName" you could register the domain name "CompanyName.com" and your e-mail address could be "UserName@CompanyName.com". Your customers would also be able to access your organization's web site by visiting "www.companyName.com" with their Web browser.

Domain Name System (DNS): A distributed database of information that is used to translate domain names, which are easy for humans to remember and use, into Internet Protocol (IP) numbers, which are what computers need to find each other on the Internet. People working on computers around the globe maintain their specific portion of this database, and the data held in each portion of the database is made available to all computers and users on the Internet. The DNS comprises computers, data files, software, and people working together.

Encoding: Encoding converts a file from one format to another so that a file can be moved across different computer systems which may not all support the same character sets. The most common form of encoding is binary-hexadecimal (Binhex) encoding. Binhex encoding is the default encoding option for any attachments that you add to messages. While encoding creates a new file which is larger than the original, it converts the data fork, resource fork, and Finder information into a character file which can easily be sent as an attachment. 4D Internet Commands support the most common encoding methods, including Binhex, Base64, AppleSingle, AppleDouble, UUEncode and MacBinary.

Encryption: Encryption is used to intentionally scramble the contents of messages. Messages are encrypted using an external encryption program such as PGP, for the sole purpose of increasing the privacy of messages. The encrypted text must then be decrypted before it can be read. 4D Internet Commands do **NOT** provide any means for encrypting text.

Compression: Is used as a means of reducing the space taken up by a file. In order to compress a file, the file must be run through an application such as Stuffit Deluxe™ Compact Pro™ or WinZip™. These files must then be decompressed using the application in order to return the file to its original format. When files are compressed using compression applications, it is common for those applications to append a suffix to the original name of the file. Below are some common suffixes and their respective applications.

Filename.SIT - Stuffit application
Filename.CPT - Compact Pro application
Filename.DD - Disk Doubler application
Filename.ZIP - Winzip application
Filename.SEA - Self Extracting Archive. These files are Macintosh stand alone applications and will decompress themselves when the user double-clicks on them because application code for decompression is included. Due to the addition of this code, self extracting archives are generally larger than if the file was created as *Filename*.SIT or *Filename*.CPT. However since the user doesn't need to have the compression application, this option may be advantageous to the end user.

It is important to remember that once compressed, a file still needs to be encoded prior to transmission to ensure that the file is properly tranferred from machine to machine on its way to its ultimate destination.

Parameter Formats

The descriptions that follow provide details on the meaning and formatting of many key parameters used throughout the 4D Internet Commands.

Parameter	Type		Description
hostName	String	→	Host name (Ex: "www.companyname.com") IP address (Ex: "204.118.90.2")
ip_LongInt	LongInt	→	Long Integer reference to a IP address
mailAddress	Text	→	Ex: "jsmith@acius.com" (do not include "<" ">" brackets sometimes seen in the examples)
addressList	Text	→	Ex: "jsmith@acius.com, jdupont@aci.fr"
localPath	Text	→	- Document Mac: "My Hard Drive:4DDB:SalesDB:Report" Win: "C:\MyDrive\4DDB\SalesDB\Report.txt" - Directory Mac: "My Hard Drive:CoolStuff:" (Note trailing ":") Win: "C:\MyDrive\CoolStuff\"
hostPath	Text	→	- Document "/usr/jsmith/reports/salesreport.txt" - Directory "/usr/jsmith/reports/"(Note trailing "/")
tcp_ID	LongInt	→	Reference to an open TCP session
smtp_ID	LongInt	→	Reference to a new mail message
pop3_ID	LongInt	→	Reference to an open POP3 session
ftp_ID	LongInt	→	Reference to an open FTP session
Function result	Integer	←	Error Code

hostName
The hostName is the host name or IP address, such as "dns.acius.com" or "204.118.90.2". Host names are resolved through a domain name system. The default and secondary domain name systems are typically set within the Control Panel of the installed TCP/IP driver. Any 4D Internet command requiring a hostName as a parameter will accept its value in either the name ("www.acius.com") or IP address ("204.118.90.2") format. The "name" format is always preferred since it buffers your application from ill-effects due to hardware changes at remote sites.

ip_LongInt
All host names can be resolved via the methods described above to an IP address. Mathematical formulas can then be applied to the IP address to convert the value to a unique long integer number. Commands within the 'Special Functions' section such as NET_NameToAddr and NET_AddrToName automate this conversion process. This LongInt value is referred to as the ip_LongInt throughout this documentation. The LongInt value will only be of use in special circumstances by developers doing direct TCP communication. Some developers may also prefer to store the LongInt value of a domain name in order to conserve disk space compared to its string equivalent.

mailAddress
The MailAddress is a fully qualified mail specification in the format "user_name@domain_name". Within this document, mailAddress refers to a **single e-mail address**. Any 4D Internet Commands parameter which can take more than one address will specifically state addressList. If a parameter has mailAddress as its only type, it can take one and only one mail address. The format of the mailAddress should be a full reference containing both the user name and domain name. The mailAddress cannot contain any comments such as those sometimes found in mail received from a POP3 server:

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

Note: Be careful to strip any comments from your address specifications when using header information obtained by POP3 calls as the basis for addressing your replies.

addressList
An addressList contains one **or more e-mail** addresses in the format of mailAddress, each delimited by a comma or carriage return. Carriage return delimiting is useful when providing users with a text field to enter or paste a number of addresses. The following three examples would each generate an acceptable $AddressList value:

```
$AddressList:="jsmith@acius.com"

$AddressList:="jsmith@acius.com, scott@acius.com, marcel@aci.fr"

For ($i;1;Size of Array(aAddresses))
   $AddressList:=$AddressList+aAddresses{$i}+Char(13)
End For
```

localPath
The localPath is the location of a file or directory on the users machine (Mac or Windows). On a Macintosh, items within folders are delimited by colons. For example, the file "My Report" in the "Reports" folder on the hard drive titled "My Hard Drive" would have a pathname of "My Hard Drive:Reports:My Report". A directory specification on a Macintosh should end with a colon character. For example, if you wanted to place a new report in the same folder as the above example, you would refer to the directory as "My Hard Drive:Reports:". The decision to reference a File or Directory name is based on the context called for by the command. A similar format is used under the Windows environment, except a backward slash "\" is used in place of the colon character.

hostPath
The hostPath is the location of a file or directory on a computer running under the Unix operating system. In the Unix environment, directories are separated with slashes ("/"). For example, the file "report.txt" in the "reports" directory in the "ACI" directory would be specified as "/ACI/reports/report.txt". A directory pathname should end with a "/" character. Note that a full pathname begins with a "/" which represents the root of the volume.

smtp_ID, **pop3_ID**, **ftp_ID**, **tcp_ID**: Throughout each section of 4D Internet Commands, references are made to an "ID" number in most of the commands. Each set of communication functions will establish their own "session" represented by a Long Integer "ID" number. Subsequent commands related to the open session will use this value to direct their effects down the proper channel.

The "ID" numbers obtained in each section (SMTP, POP3, FTP, TCP) may not be passed as values to different sections. For instance, an FTP session identified by a ftp_ID variable cannot be passed to the TCP routines for any special processing.

Session Reference	**Opened by**	**Closed by**
tcp_ID	TCP_Open or TCP_Listen	TCP_Close
smtp_ID	SMTP_New	SMTP_Close
pop3_ID	POP3_Login	POP3_Logout or POP3_VerifyID
ftp_ID	FTP_Login	FTP_Logout or FTP_VerifyID

Function result
All 4D Internet Commands (with the exception of IT_ErrorText & IT_Version) return an integer value as the result of the function. This integer contains any error number which the command needs to convey back to the 4D database.
If a command is successful, a zero will be returned. Else, an error code is returned. For more information about 4D Internet Commands error codes, please refer to Appendix C, 4D Internet Commands Error codes.

2 IT Send Mail

Sending Mail, Overview

Simple Mail Transport Protocol (SMTP) is the mail standard used throughout the internet. With 4D Internet Commands, developers can build simple mail messages with just one command, or complex messages with a series of commands. The SMTP commands enable developers to control all portions of a mail message, including Reply-To headers, Sender headers, Attachments, Comments, and References.

4th Dimension and 4D Internet Commands allow developers to create very powerful databases with the ability to send messages and attachments over the internet. Some examples of how the suite of SMTP commands could enhance your databases are:

• Automation of sending reports or documents created within 4th Dimension.
• Databases could inform developers of special occurrences (i.e. ON ERR CALL("Mail_Error"))
• Databases could execute automated mailings to people across the country

There are an unlimited number of uses for the suite of SMTP commands. These commands, combined with those for POP3 (retrieving both files and attachments), FTP, and TCP provide the 4th Dimension developer with the tools to dramatically increase the communications capabilities of their 4D databases.

Two methods of Creating a Mail Message

Within the SMTP section of commands, there are two separate methods of sending electronic mail which have previously been referred to as "simple" and "complex". The "simple" method involves a single command, SMTP_QuickSend, which accepts all the parameters necessary to address and send a message.

The majority of e-mail sent throughout the world is pretty simple in its construction, somebody "here" wants to send a "message" of some kind to somebody "there" regarding some "subject". If this were a paper letter, you would write everything up, seal and address the envelope and then take it to the post office for delivery. With SMTP_QuickSend, you can specify the "From", "To", "Subject" and "Message Body" within one command for easy e-mail delivery.

However, not all mail delivery can fit into such narrow parameters. For instance, suppose the letter above needed copies sent to other interested parties or perhaps an attachment such as your Annual Report needed to be enclosed. In these cases, photocopies of your letter would be made and reports printed as your staff collated the material and addressed the envelopes to each recipient. The SMTP commands in 4D Internet Commands simplify the electronic distribution by giving you control over all aspects of e-mail delivery. Multiple attachments, Carbon Copy & Blind Carbon Copy addressing, any mail header specification can be handled through the Built Message capabilities of the SMTP commands.

Understanding Mail Delivery

One of the critical concepts in understanding the SMTP commands relates to the method in which mail is delivered to its recipients. The SMTP commands do not directly deliver the mail to each recipient. The commands handle the proper composition and formatting of your mail and will deliver the results to the SMTP server specified by the SMTP_Host command. The SMTP server is often a machine within your own organization or at your internet service provider. The SMTP server then resolves the optimum delivery path for your mail and schedules its distribution based on settings configured by the mail administrator.

Minimum Requirements to Send a complex SMTP Message

In order to successfully deliver a mail message built using the SMTP commands, the essential commands must all be correctly defined. The following commands represent the minimum in order for e-mail delivery to be successful:

• SMTP_New
Creates the space in memory for the new message and returns a reference to be used in subsequent commands.

• SMTP_Host
Specifies the SMTP server where the message will be delivered

• SMTP_From or SMTP_Sender
At least one address in either header

• SMTP_To or SMTP_Cc or SMTP_Bcc
At least one address in one of these headers

• SMTP_Send
Sends the message

• SMTP_Clear
Clears any memory used during the creation of the message

If only the commands listed above were executed, a message would have been sent which contained no "Subject" definition and no message body. This isn't particularly useful and illustrates the need to specify additional detail in order to effectively communicate your message.

SMTP_SetPrefs

SMTP_SetPrefs (lineFeed; bodyType; lineLength) → Integer

Parameter	Type		Description
lineFeed	Integer	→	1 = [default] Add, 0 = Don't Add, -1 = No Change
bodyType	Integer	→	Body-Content-Type (1 = [default] Auto-detect, -1 = No Change)
lineLength	Longint	→	Maximum line length (0 = [default] Auto-detect, -1 = No Change)
Function result	Integer	←	Error Code

Description

The command SMTP_SetPrefs sets the preferences of messages to be sent using the SMTP commands. The command has a global scope and will effect all subsequent messages created with the SMTP commands. The configurable options effect the format of a mail message as it is sent to an SMTP server. The preference settings have an interprocess scope and effect mail creation in any 4D process.

SMTP servers recognize the end of a line to be a combined carriage return/line feed (CR/LF) character pair. This differs from most Macintosh applications which view a single carriage return as the end of line/paragraph marker.

lineFeeds is an integer value which specifies how to handle carriage returns within the body of a mail message. Passing a value of zero in this parameter will leave the message body text untouched, permitting the developer to control their own line feed additions. A value of 1 (default setting) will replace all carriage return/line feed pairings with carriage returns for you. A value of -1 will leave the current value of the preference unchanged. If you are unsure which option to choose, you should choose 1, the default value.

bodyType specifies the Body-Content-Type of the message to be sent, according to the values in the table below. If not changed, the default content type is 1 which will permit the SMTP commands to auto-detect an appropriate setting based on the contents of the message body.

-1	No change
0	Application & binary; no encoding
1	Default; will choose either "US-ASCII & 7bit" or "ISO-8859-1 & quotable-printable" based on message content.
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

lineLength specifies a maximum SMTP line length for text within the message body. The SMTP commands will "line wrap" the body text by inserting a carriage return/line feed pair at the nearest word break before the maximum line length. Any number may be specified but it is recommended that line lengths be kept below 80 characters. A value of -1 will leave the current value unchanged.

The lineLength parameter defaults to zero. A value of zero will cause the SMTP commands to use the recommended values specified within the RFC definitions for the bodyType. If the lineLength parameter is set to zero, wrapping will occur based on the following table:

Body Type	**Wrap at**
Base64	76
Quoted-Printable	76
Other...	no wrapping

Line wrapping is strongly suggested since many systems and mail programs have problems handling messages containing unlimited line lengths. Also, keep in mind that mail often travels through a number of systems before reaching its final destination and any computer along the delivery path may reject a message if it is unable to handle the message's format.

See Also

SMTP_GetPrefs.

SMTP_GetPrefs

SMTP_GetPrefs (lineFeeds; bodyType; lineLenght) → Integer

Parameter	Type		Description
lineFeeds	Integer	←	0 = Don't Add, 1 = Add LineFeeds
bodyType	Integer	←	Body-Content-Type
lineLenght	Longint	←	Maximum line length
Function result	Integer	←	Error Code

Description

The command SMTP_GetPrefs returns the current settings assigned to the SMTP preferences. The values will be at their default state unless a prior call to SMTP_SetPrefs altered the settings. For a more complete description of the parameters, see SMTP_SetPrefs.

lineFeeds returns the current setting determining how the SMTP commands will handle carriage returns within the body of a message.

bodyType returns the current setting for the Body-Content-Type. See the bodyType table under SMTP_SetPrefs for a description of the values.

lineLength returns the current setting for the maximum line length of text in the message body.

See Also

SMTP_SetPrefs.

SMTP_QuickSend

SMTP_QuickSend (hostName; msgFrom; msgTo; subject; message) → Integer

Parameter	Type		Description
hostName	String	→	Host name or IP address
msgFrom	Text	→	MailAddress or AddressList
msgTo	Text	→	MailAddress or AddressList
subject	Text	→	Subject
message	Text	→	Message
Function result	Integer	←	Error Code

Description

The command SMTP_QuickSend gives the users the ability to build and send a mail message with one command. In the event that you require greater control over your message, or the message is of a more sophisticated nature, the group of SMTP commands based on the SMTP_New command should be utilized.

hostName is the host name or IP address of the SMTP server where the message will be sent for distribution.

msgFrom is a text value containing an AddressList of one or more complete mail addresses indicating who originally sent the message. All addresses listed in the From header are visible to the recipients of the message.

msgTo contains an AddressList value of one or more complete mail addresses. The addresses identified in the msgTo header will each be sent an original copy of the message. Each recipient of the message will see any other mail addresses the message was delivered to.

subject is a text value concisely describing the topic covered in detail by the message body.

Warning: The subject of the message should not contain accentuated characters (such as é, ö, etc.).

message is a text value containing the body of the mail message. The size of the message is restricted to the 32k limit of a 4th Dimension variable or field.

Example

Here is an example of use of this command:

```
	$Host:="www.acius.com"
	$ToAddress:="adupont@aci.fr"
	$FromAddress:="jsmith@acius.com"
	$Subject:="Sales Report"
	$Message:="Can you send me the sales report for January 1999? Thanks."
⇒	$Error:=SMTP_QuickSend ($Host;$FromAddress;$ToAddress;$Subject;$Message)
	If ($Error#0)
		ALERT("Error:  SMTP_QuickSend"+ Char(13)+IT_ErrorText ($Error))
	End If
```

See Also

SMTP_New.

SMTP_New

SMTP_New (smtp_ID) → Integer

Parameter	Type		Description
smtp_ID	Longint	←	Reference to this new message
Function result	Integer	←	Error Code

Description

The command SMTP_New should be the first command called in any sequence that is going to build a SMTP mail message except where SMTP_QuickSend is being used. SMTP_New creates a new message in memory and returns a reference to the message in the smtp_ID long integer variable. Subsequent SMTP commands will use the smtp_ID reference to populate the message with header and body information prior to calling SMTP_Send.

Every call to SMTP_New should have a corresponding call to SMTP_Clear. After sending a message, the call to SMTP_Clear will free any memory held by the contents of the message.

smtp_ID is the long integer reference to the message just created. This ID will be used for all subsequent references to this message. It is possible to open multiple new messages and the smtp_ID returned for each provides a means of identifying which open message any subsequent command should be applied to.

Example

See the example for the command SMTP_Body.

See Also

SMTP_Clear, SMTP_QuickSend, SMTP_Send.

SMTP_Host

version 6.5

SMTP_Host (smtp_ID; hostName{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
hostName	String	→	Host name or IP address
deleteOption	Integer	→	0 = Add or Replace, 1 = Delete
Function result	Integer	←	Error Code

Description

All mail created and sent from the SMTP commands must be directed to a specific SMTP server. 4D Internet Commands do not deliver mail directly to each recipient, it is delivered to the SMTP server specified by this command. The SMTP server is responsible for resolving address errors and scheduling the delivery of the message.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

hostName is the host name or IP address of the SMTP server which will handle the distribution of the message.

deleteOption is an optional parameter which specifies whether to delete the current host setting. A value of zero will set the host to the value specified by hostName. A value of 1 will delete the Host specification for the message identified by smtp_ID. This is an optional parameter and will default to zero if not otherwise specified.

Example

See the example for the command SMTP_Body.

See Also

SMTP_New.

SMTP_Send

SMTP_Send (smtp_ID) → Integer

Parameter	**Type**		**Description**
smtp_ID	Longint	→	Message reference
Function result	Integer	←	Error Code

Description

The command SMTP_Send sends the message referenced by smtp_ID but does not clear the data from memory.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

Example

In this example a message is created and the static elements are defined outside the scope of the 'for' loop. Then, for each record in the [People] table, the message is customized and sent.

```
   $error:=SMTP_New ($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!")
   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)   `Replace the "To" header with new value
   $error:=SMTP_Body ($smtp_id;$Body)
⇒  $error:=SMTP_Send ($smtp_id)
   NEXT RECORD([People])
   End for
   $error:=SMTP_Clear ($smtp_id)
```

See Also

SMTP_New.

SMTP_Clear

SMTP_Clear (smtp_ID) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
		←	0 if successful
Function result	Integer	←	Error Code

Description

The command SMTP_Clear disposes of a message, freeing any memory using during its creation. Every call to SMTP_New should have a corresponding call to SMTP_Clear.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command. Upon the successful close of a SMTP message, the SMTP_Clear command will return a zero value back into the smtp_ID variable.

Example

See the example for the command SMTP_Body.

See Also

SMTP_New.

SMTP_Date (smtp_ID; msgDate; msgTime; timeZone; offset{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
msgDate	Date	→	Date this message was created
msgTime	Time	→	Time this message was created
timeZone	Integer	→	Location code
offset	Integer	→	Dependent on value in timeZone parameter
deleteOption	Integer	→	0 = Add/Replace, 1 = Delete
Function result	Integer	←	Error Code

Description

Given a date, a time, and a geographical location of the mail creator, the command SMTP_Date will build the date header for the message specified by the smtp_ID value. The date that is passed to the command should be the date and time for the current location of the machine sending the message. Since the parameters below must follow a specific format, the mail server on the receiving end of the message can interpret the date and time based on the date, time, time zone, and offset passed to it. It can then convert the sender's date and time to a local-time equivalent.

Note: If a mail message is composed without a Date header, the SMTP server will add one with its current date & time settings. All SMTP mail messages contain a date header, either added by the client application or the SMTP server.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

msgDate is a 4D date which contains the date that this message was created.

msgTime is a time which contains the time this message was created. To convert a 4D time to a long integer, add zero to the 4D time [i.e. $time:=(Current Time+0)].

timeZone identifies the time zone of the sender. This field accepts a value between zero and 6 based on the tables below.

• A value of 0 (zero) allows the user to directly specify in the offset parameter the number of hours to add or subtract from Universal Time.
• A value of 1 will have the sending machine automatically add the offset based on the Macintosh's PRAM. If the timeZone is 1 the offset parameter is not needed. The timezone of a Macintosh computer is determined by the settings in the **Map** (MacOS 8) or **Date&Time** (MacOS 8.5 and higher) control panel. Developers should give consideration to the accuracy of this option if the time values are a critical factor to their databases.

• Values 2 through 5 correspond to the 4 time zones in the United States. The offset for each of these values specify whether that time zone is in daylight saving time (offset = 1) or not (offset = 0).
• A value of 6 specifies that the time supplied will be military time. For this instance, the offset is determined by the Military Time table below. Use the corresponding offset value (-12 through 12) based on the military time code of the location of the sender.

offset - The value of this parameter is dependent upon the code set in the timeZone parameter. See the descriptions above or the table below to find the correct value to set for this parameter.

Code	**Time Zone**	**Offset Parameter**
0	+/- offset from UT	Offset is in +/- Hours
1	+/- offset from UT	Offset not used, offset is supplied by Mac's PRAM
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	Military Time	*See Table Below*

Offset Values	**Military Time Codes**
0	Z
-1 thru -9	A thru I
-10 thru -12	K thru M
1 thru 12	N thru Y

Definitions of Abbreviations

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 - A value of zero will add the date header with the given parameters, or replace a previously added set of values. A value of 1 causes any previous definition of this field to be removed. Any values in the other parameters are ignored. deleteOption is an optional parameter which will default to zero if not otherwise specified.

See Also

SMTP_New.

SMTP_From

version 6.5

SMTP_From (smtp_ID; msgFrom{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
msgFrom	Text	→	MailAddress or AddressList
deleteOption	Integer	→	0 = Add to existing list, 1 = Replace old values with the new values 2 = Remove the specified addresses
Function result	Integer	←	Error Code

Description

The command SMTP_From contains the mail address(es) of the person(s) to be listed in the "From" field of the message. The addresses in this field are those of the persons responsible for creating or authorizing the message. Normally, the "From" header contains one address representing the person who composed and sent the message. There may be circumstances however in which a message is composed by a group of people who should each be individually recognized within the "From" header.

The "From" header is not mandatory. A message can be successfully delivered which does not have a "From" header but does have an address specified within the "Sender" header. If an address is specified in the "From" header the existence of the "Sender" header is optional.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

msgFrom is a text value containing an AddressList of one or more mail addresses. All addresses listed in the From header are visible to the recipients of the message.

Auto-Reply note: If a "ReplyTo" header is not defined for the message identified by smtp_ID then all replies to the message will be directed back to each person specified in the "From" header.

deleteOption is an integer value which specifies how to handle the address(es) listed in msgFrom. A value of zero will add the new values to any previously assigned to this header. A value of 1 will replace any prior definitions with the new values. If msgFrom is a null string, all prior values will be removed and the header deleted from the message. A value of 2 will delete the specified addresses from any previously assigned values. deleteOption is an optional parameter which will default to zero if not otherwise specified

Example

In this example, three people compose a message on the subject of a company policy change which is distributed to everyone in the company. Any responses to this message would be directed back to each of the three people listed in the "From" header.

```
         $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)
```

See Also

SMTP_New.

SMTP_Sender

SMTP_Sender (smtp_ID; msgSender{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
msgSender	Text	→	MailAddress (1 only)
deleteOption	Integer	→	0 = Add/Replace, 1 = Delete
Function result	Integer	←	Error Code

Description

The command SMTP_Sender adds the e-mail address of the person that sends the message. It is intended to be used when the sender is not the actual author of the message, or to indicate who among a group of authors actually sent the message. This field is not necessary if the contents of the "Sender" field would be redundant with the "From" field.

In cases where a computer program is the creator and sender of a mail message, the Sender header should reference the mail account of the real person responsible for administering the actions of the program and not the account managed by the computer program.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

msgSender contains a single MailAddress to be listed in the Sender field of the message. Only **one** mail address may be specified for this header.

deleteOption is an integer value which specifies whether to add or delete the Sender header. A value of zero will set the Sender field to the new value, overriding any prior settings. A value of 1 will delete any address previously defined for the Sender field and remove the header from the mail envelope. deleteOption is an optional parameter which will default to zero if not otherwise specified.

Example

In this example, three executives compose a message on the subject of a company policy change which is then distributed by the secretary to everyone in the company. Any responses to this message would be directed back to each of the three people listed in the "From" header.

```
        $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)
```

See Also

SMTP_New.

SMTP_ReplyTo

SMTP_ReplyTo (smtp_ID; replyTo{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
replyTo	Text	→	MailAddress or AddressList
deleteOption	Integer	→	0 = Add to existing list, 1 = Replace old values with the new values, 2 = Remove the specified addresses
Function result	Integer	←	Error Code

Description

The command SMTP_ReplyTo provides the user with the ability to control the direction of replies made to the message. Normally, all replies to a message come back to the people it was "From". By setting the "ReplyTo" header on outgoing mail you can effect the default routing of responses to the message.

For the database developer, SMTP_ReplyTo can be very powerful tool permitting them to control the behavior of replies to automated mail. Users may want replies sent to addresses other than those listed in the From or Sender addresses, such as a separate account created to track responses.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command

replyTo is a text value containing an AddressList of one or more mail addresses. The addresses listed in this field will be used by the recipient's mail software as the default mail-account to direct their replies.

deleteOption is an integer value which specifies how to handle the address(es) listed in replyTo. A value of zero will add the new values to any previously assigned to this header. A value of 1 will replace any prior definitions with the new values. If replyTo is a null string, all prior values will be removed and the header deleted from the message. A value of 2 will delete the specified addresses from any previously assigned values. deleteOption is an optional parameter which will default to zero if not otherwise specified.

Example

In this example, 3 executives compose a message on the subject of a company policy change which is then distributed by the secretary to everyone in the company. Any responses to this message would be redirected to the secretary and "personnel_dept" and would not be seen by the executives.

```
        $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)
```

See Also

SMTP_New.

SMTP_To

SMTP_To (smtp_ID; msgTo{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
msgTo	Text	→	MailAddress or AddressList
deleteOption	Integer	→	0 = Add to existing list, 1 = Replace old values with the new values, 2 = Remove the specified addresses
Function result	Integer	←	Error Code

Description

The command SMTP_To adds the identity of the primary recipients of the message. It is not mandatory to have addresses in the To: field specifically, but there must be at least one address in either To:, Cc:, or Bcc:. All addresses listed in the "To" and "cc" headers in a mail message are visible to each recipient of the message.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

msgTo is a text value containing an AddressList of one or more mail addresses.

deleteOption is an integer value which specifies how to handle the address(es) listed in msgTo. A value of zero will add the new values to any previously assigned to this header. A value of 1 will replace any prior definitions with the new values. If msgTo is a null string, all prior values will be removed and the header deleted from the message. A value of 2 will delete the specified addresses from any previously assigned values. deleteOption is an optional parameter which will default to zero if not otherwise specified.

Example

See the example for the command SMTP_Body.

See Also

SMTP_New.

SMTP_Cc

SMTP_Cc (smtp_ID; carbonCopy{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
carbonCopy	Text	→	MailAddress or AddressList
deleteOption	Integer	→	0 = Add to existing list, 1 = Replace old values with the new values, 2 = Remove the specified addresses
Function result	Integer	←	Error Code

Description

The command SMTP_Cc adds carbon copy recipients to the message specified by smtp_ID. It is not mandatory to have any addresses in the Cc: field, but there must be at least one address in either To:, Cc:, or Bcc:. All addresses listed in the "To" and "cc" headers in a mail message are visible to each recipient of the message.

smtp_ID is the long integer reference to a message created with the SMTP_New command.

carbonCopy is a text value containing an AddressList of one or more mail addresses.

deleteOption is an integer value which specifies how to handle the address(es) listed in carbonCopy. A value of zero will add the new values to any previously assigned to this header. A value of 1 will replace any prior definitions with the new values. If carbonCopy is a null string, all prior values will be removed and the header deleted from the message. A value of 2 will delete the specified addresses from any previously assigned values. deleteOption is an optional parameter which will default to zero if not otherwise specified.

Example

See the example for the command SMTP_Body.

See Also

SMTP_Bcc, SMTP_New.

SMTP_Bcc

SMTP_Bcc (smtp_ID; blindCarbon{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
blindCarbon	Text	→	AddressList
deleteOption	Integer	→	0 = Add to existing list, 1 = Replace old values with the new values, 2 = Remove the specified addresses
Function result	Integer	←	Error Code

Description

The command SMTP_Bcc adds blind carbon copy recipients to the message specified by smtp_ID. It is not mandatory to have any addresses in the Bcc: field, but there must be at least one address in either To:, Cc:, or Bcc:.

The only way to keep AddressList information confidential when sending mail to groups of people is to list the addresses within the "Bcc" header. The addresses listing in a "Bcc" header are not sent as part of the message header or body. The addresses will not be viewable by any recipient specified in the "To", "Cc" or "Bcc" headers.

The "Bcc" recipients will be able to see all "To" and "Cc" recipients, but they will not be able to see other "Bcc" recipients. Often group mailings to a large number of recipients will be addressed with all recipients placed in the "Bcc" header. This prevents the users from having large address lists cluttering the message and keeps them from accessing the addresses of others.

Another reason for the use of "Bcc" is that many mail applications have a "Reply All" feature which will add all recipients in the "To" and "Cc" sections to the replying message. Placing all recipient addresses within the "Bcc" header will prevent users from replying to every person who received the original message.

smtp_ID is the long integer reference to a message created with the SMTP_New command.

blindCarbon is a text value containing an AddressList of one or more mail addresses.

deleteOption is an integer value which specifies how to handle the address(es) listed in blindCarbon. A value of zero will add the new values to any previously assigned to this header. A value of 1 will replace any prior definitions with the new values. If blindCarbon is a null string, all prior values will be removed and the header deleted from the message. A value of 2 will delete the specified addresses from any previously assigned values. deleteOption is an optional parameter which will default to zero if not otherwise specified.

Example

In this example a message is created and the static elements are defined outside the scope of the 'for' loop. Then, for each record in the [People] table, an addresses is added to the blind carbon copy list.

```
        $error:=SMTP_From ($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)  `Add this email address to the BCC list
            NEXT RECORD([People])
        End for
        $error:=SMTP_Send ($smtp_id)   `Send the message to everyone
        $error:=SMTP_Clear ($smtp_id)
```

See Also

SMTP_Cc, SMTP_New.

SMTP_InReplyTo

SMTP_InReplyTo (smtp_ID; inReplyTo{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
inReplyTo	Text	→	In-Reply-To Text
deleteOption	Integer	→	0 = Add/Replace, 1 = Delete
Function result	Integer	←	Error Code

Description

The command SMTP_InReplyTo identifies the previous correspondence which this message is a response to.

smtp_ID is thelong integer reference to the mail message created with the SMTP_New command.

inReplyTo is a text value which references previous correspondences to which this message pertains. For specific formatting requirements, please consult RFC#822.

Warning: The text should not contain a line feed (ascii=10). Doing so would signify the end of the header section and the beginning of the body. Subsequent header items could be pushed into the body and not recognized properly by the server or client software. For more information regarding the headers, please refer to RFC#822.

deleteOption is an optional integer which specifies whether to delete any inReplyTo text. A value of zero will add the text to the inReplyTo field. A value of 1 will delete any text from the inReplyTo field, and will ignore any text in inReplyTo.

See Also

SMTP_New.

SMTP_References

SMTP_References (smtp_ID; references{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
references	Text	→	Reference Text
deleteOption	Integer	→	0 = Add/Replace, 1 = Delete
Function result	Integer	←	Error Code

Description

The command SMTP_References identifies additional correspondences that the message references.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

references is a text value containing the reference text. For specific formatting requirements, please consult RFC#822.

Warning: The text should not contain a line feed (ascii=10). Doing so would signify the end of the header section and the beginning of the body. Subsequent header items could be pushed into the body and not recognized properly by the server or client software. For more information regarding the headers, please refer to RFC#822.

deleteOption is an optional integer which specifies whether to delete the references header. A value of zero will add the text to the references field. Any text previously in references will be replaced. A value of 1 will delete all text from the references field, regardless of any text listed in references.

See Also

SMTP_New.

SMTP_Comments

SMTP_Comments (smtp_ID; comments{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
comments	Text	→	Comment text
deleteOption	Integer	→	0 = Add/Replace, 1 = Delete
Function result	Integer	←	Error Code

Description

The command SMTP_Comments lets the user add text comments to the message while keeping the message's body untouched. The comments only appear within the header section of the message. Many mail readers do not display the full text of the message header to their users.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

comments is a text value containing information you would like placed in the mail header.

Warning: The text should not contain a line feed (ascii=10). Doing so would signify the end of the header section and the beginning of the body. Subsequent header items could be pushed into the body and not recognized properly by the server or client software. For more information regarding the headers, please refer to RFC#822.

deleteOption specifies whether to set or remove the comments text. A value of zero will define the comments header and overwrite any value previously set for the field. A value of 1 will remove the comments header and will ignore any values in comments. deleteOption is an optional parameter which will default to zero if not specified.

Example

See the example for the command SMTP_Body.

See Also

SMTP_New.

SMTP_Keywords

SMTP_Keywords (smtp_ID; keywords{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
keywords	Text	→	Keywords List
deleteOption	Integer	→	0 = Add, 1 = Delete
Function result	Integer	←	Error Code

Description

This field contains keywords or phrases, separated by commas.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

keywords is a text value which contains a keyword or keyword list. For specific formatting requirements, please consult RFC#822.

Warning: The text should not contain a line feed (ascii=10). Doing so would signify the end of the header section and the beginning of the body. Subsequent header items could be pushed into the body and not recognized properly by the server or client software. For more information regarding the headers, please refer to RFC#822.

deleteOption is an optional integer value which specifies whether to delete the current keywords list. A value of zero will append the text in keywords to any text already added. A value of 1 will delete the current value in keywords and replace it with that in keywords. In this case, a null string will delete the keywords header entirely.

See Also

SMTP_New.

SMTP_Encrypted

version 6.5

SMTP_Encrypted (smtp_ID; encrypted{; deleteOption})

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
encrypted	Text	→	Encryption method
deleteOption	Integer	→	0 = Add/Replace, 1 = Delete

Description

The command SMTP_Encrypted informs the user of the type of encryption used on the body of the message. 4D Internet Commands **do not** provide the ability to encrypt or decrypt mail messages. The encryption of a message body is left to the developer. If steps are taken to encrypt the message body (prior to its assignment via SMTP_Body), this command should be used to identify the encryption method employed.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

encrypted is a text value which specifies the type of encryption method used to encrypt the body of the message. The encrypted header is used by the recipients mail software to determine the method needed to decrypt the message body. For specific formatting requirements, please consult RFC#822.

Warning: The text should not contain a line feed (ascii=10). Doing so would signify the end of the header section and the beginning of the body. Subsequent header items could be pushed into the body and not recognized properly by the server or client software. For more information regarding the headers, please refer to RFC#822.

deleteOption is an optional integer which specifies whether or not to delete the header which contains the text of the encryption method. A value of zero will add the header with the corresponding text. Any text previously in encrypted will be replaced. A value of 1 will remove the "Encrypted" header from the message.

See Also

SMTP_Body, SMTP_New.

SMTP_AddHeader

SMTP_AddHeader (smtp_ID; headerName; headerText{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
headerName	String	→	Name of header
headerText	Text	→	Header text
deleteOption	Integer	→	0 = Add 1 = Replace all headers with 'headerName', 2 = Remove all headers named 'headerName'
Function result	Integer	←	Error Code

Description

The command SMTP_AddHeader allows users to add their own headers to the message referenced by smtp_ID. Beyond the various headers 4D Internet Commands have provided commands for, there are two additional categories of headers, these being 'user-defined' and 'extended' headers. The SMTP_AddHeader command permits the user to add both the new header tag and the data to associate with it.

Extended-Headers: These headers have been officially recognized by the NIC and were defined after the original SMTP specification. These headers often have a specific function to effect behavior in various software applications. Extended headers never begin with the letter "X".

User-Defined Headers: The SMTP protocol allows anyone to create their own header definitions. All user-defined headers should begin with the characters "X-" so there will be no possibility of conflict with a future Extended-Header. User-defined headers are tremendously useful when you have design control over both ends of the communications.

User defined headers allow the developer to store data which can easily be pulled out using the POP3 external command POP3_FindHeader. For example, you may create a header named "X-001001", which contains the value in field 01 of file 01. An unlimited number of headers may be added to a message. User-defined headers give the user the ability to add information which can easily be extracted without the need to parse through the body of the message to find the appropriate information.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

headerName is a string which contains the name of the header to be added.

headerText is a text value containing the information to be assigned to the field identified by headerName.

Warning: The text should not contain a line feed (ascii=10). Doing so would signify the end of the header section and the beginning of the body. Subsequent header items could be pushed into the body and not recognized properly by the server or client software. For more information regarding the headers, please refer to RFC#822.

deleteOption is an optional integer parameter which specifies whether to delete the current header. A value of zero will add headerName to the message. A value of 1 will replace all headers with headerName. In this case, if headerName is a null string, all headers will be removed. A value of 2 will remove all headers named headerName.

Example

See the example for the command SMTP_Body.

See Also

POP3_FindHeader, SMTP_New.

SMTP_Subject

SMTP_Subject (smtp_ID; subject{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
subject	Text	→	Subject of message
deleteOption	Integer	→	0 = Add/Replace, 1 = Delete
Function result	Integer	←	Error Code

Description

The command SMTP_Subject adds the subject of the message to the message referenced by smtp_ID. If a subject has already been added by a previous SMTP_Subject command, the new subject will override the previous subject.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

subject is a text value concisely describing the topic covered in detail by the message body.

Warning: The subject of the message should not contain accentuated characters (such as é, ö, etc.).

Warning: The text should not contain a line feed (ascii=10). Doing so would signify the end of the header section and the beginning of the body. Subsequent header items could be pushed into the body and not recognized properly by the server or client software. For more information regarding the headers, please refer to RFC#822.

deleteOption is an optional integer value specifying whether to delete the subject of the message referenced by smtp_ID. A value of zero will set the subject to the provided text string. A value of 1 will delete the Subject header from the message and leave it blank (essentially ignoring anything passed in the subject parameter).

Example

See the example for the command SMTP_Body.

See Also

SMTP_New.

SMTP_Body

SMTP_Body (smtp_ID; msgBody{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
msgBody	Text	→	Body of message
deleteOption	Integer	→	0 = Add/Replace, 1 = Delete, 2 = Append
Function result	Integer	←	Error Code

Description

The command SMTP_Body assigns the text in msgBody to the main body section of the mail message identified by smtp_ID. The msgBody is the main block of text

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

msgBody is a text value which contains the body of the message. The size of msgBody is restricted to the 32K limit of a 4D text object. This does not mean that the mail message itself has a 32K limit. In order to send a letter whose body is greater than 32K, you must use the Append flag of the deleteOption parameter (see below). The actual size limitation of a mail message body is limited only by available memory.

deleteOption is an optional integer value specifying whether to delete the body of the message referenced by smtp_ID. A value of zero will set the body to the provided text string. A value of 1 will delete the body from the message and leave it blank (essentially ignoring anything passed in the msgBody parameter). A value of 2 will append the text in msgBody to any text that had already been sent by a previous call to SMTP_Body.

Example

Here is a complete SMTP example:

```
C_LONGINT($SMTP_ID)
C_BOOLEAN($SentOK;$OK)
$SentOK:=False  `A flag to indicate if we made it through all of the commands
Case of
   : (Not(ERRCHECK ("SMTP_New";SMTP_New ($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-ACIdemo:";◊VERSION))))
⇒  : (Not(ERRCHECK ("SMTP_Body";SMTP_Body ($SMTP_ID;vMessage))))
   : (Not(ERRCHECK ("SMTP_Send";SMTP_Send ($SMTP_ID))))
Else
   $SentOK:=True  `message was composed and mailed successfully
End case
If ($SMTP_ID#0)  `If a Message Envelope was created we should clear it now
   $OK:=ERRCHECK ("SMTP_Clear";SMTP_Clear ($SMTP_ID))
End if
```

Below is the code for the method *ERRCHECK*. This method takes two parameters, the name of the command ($Command), and the error value (passed by executing the command in the parameter of the method. *ERRCHECK* returns a boolean value corresponding to whether the error was zero. If the error is not zero, the return value ($0) gets false, otherwise it is true.

```
C_TEXT(vErrorMsg)
$Command:=$1
$Error:=$2
$Result:=True
If ($Error#0)
   $Result:=False
   If (◊SHOWERRORS)  `boolean to determine whether to display error messages
      vErrorMsg:=IT_ErrorText ($Error)
      ALERT("ERROR ---"+Char(13)+"Command: "+$Command+Char(13)+
                 "Error »»Code:"+String($Error)+Char(13)+"Description: "+vErrorMsg)
   End if
End if
$0:=$Result
```

See Also

SMTP_New.

SMTP_Attachment

SMTP_Attachment (smtp_ID; fileName; encodeType{; deleteOption}) → Integer

Parameter	Type		Description
smtp_ID	Longint	→	Message reference
fileName	Text	→	Name of file to attach
encodeType	Integer	→	0 = No encoding (sends DataFork only) ±1 = BinHex ±2 = Base64; (sends DataFork only) ±3 = AppleSingle ±4 = AppleDouble ±5 = AppleSingle AND Base64 ±6 = AppleDouble AND Base64 ±7 = UUEncode ±8 = MacBinary
deleteOption	Integer	→	0 = Add to existing list, 1 = Replace all attachments with Filename, 2 = Remove only this attachment
Function result	Integer	←	Error Code

Description

The command SMTP_Attachment provides a means to attach text or binary files to your message in MIME format. This command may be called multiple times in order to attach multiple documents to one mail message. If a value greater than zero is passed to the EncodeType parameter, this command will perform encoding at the time the message is sent.

smtp_ID is the long integer reference to the mail message created with the SMTP_New command.

fileName contains the name of the file you want to attach to the message. This value may be specified three different ways:

""	= Display the Standard Open File dialog.
"FileName"	= Looks for FileName in the same directory as the structure of the database.
"Path:FileName"	= Complete path of the file including FileName.

encodeType is an Integer value indicating what type of encoding will be done on the file before it is incorporated into the message. If attaching a binary file, an encoding method must be applied capable of the proper conversion (BinHex, AppleSingle). The most common encoding method is BinHex.

If you pass positive values of encodeType the command will automatically encode the file using the specified method when the message is sent. The encoding of a file occurs at the time the SMTP_Send command is issued. If the file is large it may take some time for the SMTP_Send command to complete. Significant time may be saved in cases where the same file will be sent a number of times. In these cases it is best to encode the file one time with the IT_Encode command and then attach the resulting file to your message using the negative value of encodeType. A negative value in encodeType will not perform any additional encoding but will set the message headers to the correct encoding method of the attached file. This will inform your recipients' mail reader of the correct way to interpret your attachment.

deleteOption is an optional integer parameter which specifies how to treat the attachment. A value of zero will add the attachment to the current list of attachments. A value of 1 will replace all attachments with the file in fileName. If fileName is a null string, all attachments will be removed. A value of 2 will remove only the attachment listed in fileName from the list of attachments.

See Also

IT_Encode, SMTP_New, SMTP_Send.

3 IT Review Mail

The suite of POP3 commands enable your database to retrieve messages from a POP3 mail server. 4D Internet Commands are MIME compliant and can recognize and extract messages containing multiple enclosures.

The POP3-related commands are broken down into two sections, "POP3 - IT Review Mail" and "POP3 - IT Downloaded Mail". The separation of commands is representative of the differing methods for reading mail. When reading mail from a POP3 server, messages (or information about the messages) may be brought down into 4D structures (variables, fields, arrays) or they may be downloaded to disk. This section "POP3-IT Review Mail" covers the ability of 4D Internet Commands to read messages from the POP3 server into 4D.

The need to for dual methods of message retrieval is spawned by memory constraints on actions which have the potential to download many megabytes of information. For instance, a single mail message which had a 5 Mb attachment could easily overflow the storage capability within the database. The only 4D structure capable of storing this size is a picture or a BLOB field, but converting a message or attachment to this format is often ineffectual since it places huge memory requirements on any client attempting to access the picture or the BLOB. To resolve this issue, this section has a command POP3_Download which will bring a message from the POP3 server to the user's local disk. Once on disk, the next section of the manual provides a number of commands to manipulate the file.

When using the suite of POP3 commands, it is important to understand the parameters that are used most frequently, especially msgNumber and uniqueID. msgNumber is the number of a message in the mailbox at the time the POP3_Login command was executed. Upon login, messages in a mailbox are assigned numbers from 1 to the number of items in the mailbox. Numbers are assigned based on the order that they were received in the mailbox, with one being the oldest message. The numbers assigned to the messages are only valid during the time from your POP3_Login to POP3_Logout.

At the time POP3_Logout is executed any message marked for deletion will be removed. When the user logs back into the server, the current messages in the mailbox will once again be numbered from 1 to x. For example, if there are 10 messages in the mailbox, and messages numbered 1 through 5 are deleted, messages 6 through 10 will be renumbered 1 through 5 the next time the user logs in to the mailbox.

To illustrate, suppose you login to a POP3 server and obtain the following list of messages:

#	UniqueID	Date	From	Subject
1	bd573a4dbd573a4d	1 Jul 1998 ...	danw@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

During the session you delete message numbers 3 and 4. When you Logout of this session your requested deletions are committed. Now when you log back into the server your message list would be renumbered as:

#	UniqueID	Date	From	Subject
1	bd573a4dbd573a4d	1 Jul 1998 ...	danw@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 is not a static value in relation to any specific message and will change from session to session dependent on its relation to other messages in the mailbox at the time the session was opened. The uniqueID however is a unique number assigned to the message when it was received by the server. This number is calculated using the time and date that the message is received and is a value assigned by your POP3 server. Unfortunately, POP3 servers do not use the uniqueID as the primary reference to its messages. Throughout the POP3 commands you will need to specify the msgNumber as the reference to messages on the server. Developers may need to take some care if developing solutions which bring references to messages into a database but leave the body of the message on the server.

POP3_SetPrefs

POP3_SetPrefs (stripLineFeed; msgFolder; attachFolder) → Integer

Parameter	Type		Description
stripLineFeed	Integer	→	0 = Don't Strip LineFeeds, 1 = Strip LineFeeds, -1 = No Change
msgFolder	Text	→	Messages folder path ("" = no change)
attachFolder	Text	→	Attachments folder path ("" = no change)
Function result	Integer	←	Error Code

Description

The command POP3_SetPrefs sets the preferences for all POP3 commands.

stripLineFeed is an integer value specifying how LineFeed characters will be treated in saved messages. Most POP3 servers combine Carriage Return and Line Feed characters to indicate the end of a line. Macintosh applications prefer a carriage return only as the end-of-line character. This option lets users strip the linefeed character from their message text. A value of zero will leave retrieved messages in the format as stored on the POP3 server. A value of 1 will strip linefeed characters from retrieved messages. A value of -1 will leave this preference as it has been previously set. The default option defaults to 1 and will automatically strip linefeeds found in messsages.

msgFolder is a text value indicating the local pathname to a folder in which messages retrieved with the POP3_Download command are stored by default.

attachFolder is a text value containing the local pathname to a folder in which attachments are stored when the MSG_Extract command separates the attachments from the main body of a message.

See Also

MSG_Extract, POP3_Download, POP3_GetPrefs.

POP3_GetPrefs (stripLineFeed; msgFolder; attachFolder) → Integer

Parameter	Type		Description
stripLineFeed	Integer	←	0 = Don't Strip CR/LF, 1 = Strip CR/LF
msgFolder	Text	←	Messages folder path ("" = no change)
attachFolder	Text	←	Attachments folder path ("" = no change)
Function result	Integer	←	Error Code

Description

The command POP3_GetPrefs returns the current preferences for the POP3 commands. The preferences are returned into the variables listed in the parameters.

stripLineFeed returns the current setting of the users preference for linefeed stripping.

msgFolder is a text variable which returns the local pathname to the default folder in which retrieved messages are stored.

attachFolder is a text variable which returns the local pathname to the default folder in which extracted attachments are stored.

See Also

POP3_SetPrefs.

POP3_Login

IT Review Mail

version 6.5

POP3_Login (hostName; userName; password; aPOP; pop3_ID) → Integer

Parameter	Type		Description
hostName	String	→	Host Name or IP address of the POP3 mail server
userName	String	→	User name
password	String	→	Password
aPOP	Integer	→	0 = Cleartext Login, 1 = APOP Login
pop3_ID	Longint	←	Reference to this POP3 login
Function result	Integer	←	Error Code

Description

The command POP3_Login logs the user into the POP3 mail server with the given userName and password. If aPOP is 1 then the APOP mechanism (RFC#1321) is used to login. If aPOP is zero or not given then a normal cleartext password login is performed. The particular login is given a reference (pop3_ID) which subsequent commands can refer to.

Warning: POP3 servers were not designed to be accessed in an interactive fashion. Once you have logged in to a server you should perform whatever actions are needed and then log out of the server as soon as possible. Between your calls of POP3_Login and POP3_Logout, your procedure should not sit in any user-interactive screen. A POP3 server will automatically disconnect any sessions which do not show activity for a certain period of time. According to the RFC for POP3, the inactivity timer is supposed to be a minimum of 30 minutes. However, our experience has shown that most servers force inactive clients out after a much shorter period of time.

Each command that interacts with the POP3 server will force a reset of your inactivity timer. In the event that the server aborts your connection before you have issued a POP3_Logout call, any deletions you had performed would be rolled back.

hostName is the host name or IP address of the POP3 mail server. It is recommended that the host name be used, but if needed an IP address can be used.

userName is the user's name on the POP3 mail server. The userName should not contain the domain. For example, for the address "jack@acius.com", userName would be just "jack".

password is the password for userName on the POP3 mail server.

aPOP is an integer value indicating whether the APOP mechanism is used to login. A value of 1 will use the APOP mechanism. A zero value will perform a cleartext password login. The default value is zero.

pop3_ID is a long integer variable into which is returned a reference to the session just established. The variable will be used in all subsequent commands which perform actions related to this session.

See Also

POP3_Logout.

POP3_VerifyID

IT Review Mail

version 6.5

POP3_VerifyID (pop3_ID) → Integer

Parameter	Type		Description
pop3_ID	Longint	→	Reference to a POP3 login
		←	0 = Connection has already closed
Function result	Integer	←	Error Code

Description

A POP3 server will automatically terminate sessions which do not show activity in a period of time determined by its administrator. Each command that interacts with the POP3 server forces a reset of the inactivity timer. The POP3_VerifyID command resets the inactivity time for the specified POP3 session without performing any other action. This allows the user to keep a session active if the possibility exists that the session may timeout.

When executed, the POP3_VerifyID command will verify the connection has not already been closed. If the session is still open the command will tell the POP3 server to reset the timeout counter for the session back to zero. If the connection has already closed, POP3_VerifyID will return the appropriate error and free memory used by the POP3 session, and return a zero value back to pop3_ID.

pop3_ID is a long integer reference to an open session created with POP3_Login.

See Also

POP3_Login.

POP3_Reset

IT Review Mail

version 6.5

POP3_Reset (pop3_ID) → Integer

Parameter	Type		Description
pop3_ID	Longint	→	Reference to a POP3 login
Function result	Integer	←	Error Code

Description

The command POP3_Reset resets the high message count and undeletes any messages marked as deleted during the current session.

Note: The POP3_Delete command only sets a flag for messages to be deleted. Messages on a POP3 server are only deleted at the time of a successful logout (POP3_Logout).

pop3_ID is a long integer reference to an open session created with POP3_Login.

See Also

POP3_Delete, POP3_Login.

POP3_Delete

POP3_Delete (pop3_ID; startMsg; endMsg) → Integer

Parameter	Type		Description
pop3_ID	Longint	→	Reference to a POP3 login
startMsg	Longint	→	Starting message number
endMsg	Longint	→	Ending message number
Function result	Integer	←	Error Code

Description

Given a range of messages from startMsg to endMsg, the command POP3_Delete will mark each message to be deleted. The act of deleting the message does not occur until you successfully issue the POP3_Logout command. If your current session terminates for any reason (timeout, network failure, etc.) prior to calling the POP3_Logout command, any messages marked for deletion will remain on the POP3 server.

pop3_ID is a long integer reference to an open session created with POP3_Login.

startMsg is a long integer number which is the starting message number of the messages to delete.

endMsg is a long integer number which is the ending message number of the messages to delete.

Note: The POP3_Delete, POP3_MsgLstInfo and POP3_MsgLst commands do not return an error if the startMsg is greater than the endMsg. In the event that this occurs, this command – in effect – does nothing.

See Also

POP3_Logout.

POP3_Logout

POP3_Logout (pop3_ID) → Integer

Parameter	Type		Description
pop3_ID	Longint	→	Reference to a POP3 login
		←	0 = Command successfully logs off
Function result	Integer	←	Error Code

Description

The command POP3_Logout will log out of the open POP3 session referred to by the pop3_ID variable. If the command successfully logs off the POP3 server a zero value is returned back as the current pop3_ID.

Logging out from a POP3 server will signal the server that you wish to commit any deletions you made during that session. To rollback any deletions you may have made prior to logout, use the POP3_Reset command prior to POP3_Logout.

pop3_ID is a long integer reference to an open session created with POP3_Login.

See Also

POP3_Reset.

POP3_BoxInfo

IT Review Mail

version 6.5

POP3_BoxInfo (pop3_ID; msgCount; msgSize) → Integer

Parameter	Type		Description
pop3_ID	Longint	→	Reference to a POP3 login
msgCount	Longint	←	Number of messages
msgSize	Longint	←	Size of all messages
Function result	Integer	←	Error Code

Description

The command POP3_BoxInfo returns information about number and size of messages currently in the mailbox of the open session referenced by pop3_ID.

pop3_ID is a long integer reference to an open session created with POP3_Login.

msgCount is a long integer value returned containing the number of messages in the mailbox.

msgSize is a long integer value returned containing the total size of all messages in the mailbox.

See Also

POP3_Login.

POP3_MsgInfo

POP3_MsgInfo (pop3_ID; msgNumber; msgSize; uniqueID) → Integer

Parameter	Type		Description
pop3_ID	Longint	→	Reference to a POP3 login
msgNumber	Longint	→	Message number
msgSize	Longint	←	Message size
uniqueID	String	←	Unique ID of message on server
Function result	Integer	←	Error Code

Description

The command POP3_MsgInfo returns information about the message identified by msgNumber within the open mailbox referenced by pop3_ID. Information about the size of the message and its Unique ID will be returned.

pop3_ID is a long integer reference to an open session created with POP3_Login.

msgNumber is a long integer value indicating which message in the mailbox you wish to retrieve information about. The msgNumber represents the position of a message within the current list of messages. You cannot rely on the msgNumber remaining the same for a specific e-mail item from session to session.

msgSize is the long integer value returned containing the size of the message referenced by msgNumber.

uniqueID is a string variable denoting the Unique ID of the message on the server. The uniqueID is a value assigned to the message by the POP3 server software. This value will not change from session to session in the same way as msgNumber. The uniqueID value is a good reference to verify if your database has already downloaded a message from the server.

See Also

POP3_Login.

POP3_GetMessage

IT Review Mail

version 6.5

POP3_GetMessage (pop3_ID; msgNumber; offset; lenght; msgText) → Integer

Parameter	Type		Description
pop3_ID	Longint	→	Reference to a POP3 login
msgNumber	Longint	→	Message number
offset	Longint	→	Offset of character at which to begin retrieval
lenght	Longint	→	How many characters to return
msgText	Text	←	Message Text
Function result	Integer	←	Error Code

Description

The command POP3_GetMessage returns the complete text of the message identified by msgNumber within the mailbox referenced by pop3_ID. Unless otherwise specified by the POP3_SetPrefs command, any linefeed characters within the message will be removed. The POP3_GetMessage command returns the entire block of the message, including header information.

pop3_ID is a long integer reference to an open session created with POP3_Login.

msgNumber is a long integer value indicating which message in the mailbox to retrieve. The msgNumber represents the position of a message within the current list of messages. You cannot rely on the msgNumber remaining the same for a specific e-mail item from session to session.

offset is a long integer value indicating the number of characters from the beginning of the message to begin reading. In most circumstances a zero should be passed to this parameter.

length is a long integer value representing the number of characters beyond the offset position to retrieve. Since the maximum length of a 4D text variable or field is limited to 32K (32767 characters), the length parameter should be set to any number below 32K. Messages whose size is greater than 32K must be retrieved to disk via the POP3_Download command.

msgText is a text variable or field which will receive the retrieved text.

See Also

POP3_Download, POP3_SetPrefs.

POP3_MsgLstInfo

IT Review Mail

version 6.5

POP3_MsgLstInfo (pop3_ID; startMsg; endMsg; sizeArray; msgNumArray; idArray) → Integer

Parameter	Type		Description
pop3_ID	Longint	→	Reference to a POP3 login
startMsg	Longint	→	Start message number
endMsg	Longint	→	End message number
sizeArray	Longint Array	←	Array of sizes
msgNumArray	Longint Array	←	Array of message numbers
idArray	Str | Txt Array	←	Array of Unique ID's
Function result	Integer	←	Error Code

Description

The command POP3_MsgLstInfo returns information about a set of messages in a mailbox. The information is returned into three arrays with each element of the arrays corresponding to one message. Information is returned about the size of each message, the message number, and the Unique-ID of the message. The arrays passed as parameters must be of pre-declared types, though they may be of any size. The POP3_MsgLstInfo command will reset the size of each array to number of messages retrieved.

The POP3_MsgLstInfo command will not return an error number if it fails to retrieve information on any message within the current message list. If an error is encountered, no element is created in the arrays for the problem message. If the command reads each message successfully, the msgNumArray should contain numeric values in a sequential order. If problems were encountered, there may be gaps in the sequence of numbers held in msgNumArray.

pop3_ID is a long integer reference to an open session created with POP3_Login.

startMsg is a long integer number which specifies the starting message number of the message range to be examined. The message number is a value representing the position of a message within the list of all messages in the mailbox identified by pop3_ID.

endMsg is a long integer number which specifies the ending message number of the message range to be examined. The message number is a value representing the position of a message within the list of all messages in the mailbox identified by pop3_ID.

sizeArray is a long integer array returned containing the sizes of each message between startMsg and endMsg.

msgNumArray is a long integer array returned containing the message numbers between startMsg and endMsg.

idArray is a string or text array returned containing the Unique-ID's of the messages between startMsg and endMsg.

Note: The POP3_Delete, POP3_MsgLstInfo and POP3_MsgLst commands do not return an error if the startMsg is greater than the endMsg. In the event that this occurs, this command – in effect – does nothing.

See Also

POP3_MsgInfo, POP3_MsgLst.

POP3_MsgLst (pop3_ID; start; end; hdrArray; msgNumArray; idArray; valueArray) → Integer

Parameter	Type		Description
pop3_ID	Longint	→	Reference to a POP3 login
start	Longint	→	Start message number
end	Longint	→	End message number
hdrArray	Str | Txt Array	→	Array of Headers to retrieve
msgNumArray	Longint Array	←	Array of message numbers
idArray	String Array	←	String array of Unique ID's
valueArray	2D Str | Txt Array	←	2D Array of header values
Function result	Integer	←	Error Code

Description

The command POP3_MsgLst is used to get specific information of mailbox contents. hdrArray is a string or text array which lists the specific mail headers you wish to retrieve. valueArray is a 2-dimensional array which receives the data for each header specified in hdrArray. Each requested header will have a corresponding array in the first dimension of valueArray.

This command allows the user to request specific columns of the message list. This command can only return values of header items, it cannot be used to retrieve the body of a message.

Example

```
        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@acius.com"
        aValues{3}{1} may equal "Call your wife"
```

Errors are handled in the following manner:

1) Only communication-related error codes will be returned. If the command can't complete its task because of an error (network, syntax, server, etc.) then the appropriate error code will be returned.

2) If a message within the specified range of messages does not exist or gets an error:
-- No array element is created for that message.
-- No error code will be returned

3) The inability to locate any or all of the specified headers within any message does not constitute an error:
-- An array element for the message will be created
-- The Message Number and UniqueID array element will contain the appropriate values
-- For each header which does not exist in the message, a null string will be returned into that array element
-- No error code will be returned

Note: The POP3_Delete, POP3_MsgLstInfo and POP3_MsgLst commands do not return an error if the startMsg is greater than the endMsg. In the event that this occurs, this command – in effect – does nothing.

See Also

POP3_MsgInfo, POP3_MsgLstInfo.

POP3_DownLoad

POP3_DownLoad (pop3_ID; msgNumber; headerOnly; fileName) → Integer

Parameter	Type		Description
pop3_ID	Longint	→	Reference to a POP3 login
msgNumber	Longint	→	Message number
headerOnly	Integer	→	0 = Entire message, 1 = Only header
fileName	Text	←	Local Filename
Function result	Integer	←	Error Code

Description

The command POP3_DownLoad is designed to retrieve a message from a POP3 server by downloading it to a disk-based file. Any POP3 message which contains attachments or whose size is greater than 32K should be downloaded with this command. Attachments to a message can only be extracted from messages retrieved in this way.

pop3_ID is a long integer reference to an open session created with POP3_Login.

msgNumber is a long integer value indicating which message in the mailbox to retrieve. The msgNumber represents the position of a message within the current list of messages. You cannot rely on the msgNumber remaining the same for a specific e-mail item from session to session.

headerOnly is an integer value which denotes whether to retrieve the entire contents of the message or just the header information.

fileName contains the name of the file and the optional path where you would like the message saved. This value may be specified three different ways:

""	= Saves the file in the folder set by POP3_SetPrefs, with the name "temp1" (if a file with the same name already exists, the filenames "temp2", "temp3", etc. will be tried until an unused file name is found)
"FileName"	= Saves the file in the folder set by POP3_SetPrefs, titled fileName
"Path:FileName"	= Saves the file in the path specified with the name fileName

In the first two cases, if no folder has been specified by POP3_SetPrefs, the message will be saved in the same folder as the structure of the database. After the file has been saved to disk, the final name of the file will be returned to the variable passed as the fileName parameter. If you attempt to call POP3_Download with a fileName that already exists within the download folder, the name will be numerically incremented and its new value as saved to disk will be returned to the fileName variable.

See Also

POP3_SetPrefs.

POP3_UIDToNum

POP3_UIDToNum (pop3_ID; uniqueID; msgNumber) → Integer

Parameter	Type		Description
pop3_ID	Longint	→	Reference to a POP3 login
uniqueID	String	→	Unique ID of message on server
msgNumber	Longint	←	Message number
Function result	Integer	←	Error Code

Description

The command POP3_UIDToNum converts a message's Unique ID value to its **current** msgNumber within the list of messages in the mailbox referenced by pop3_ID. Since a specific mail message's msgNumber is a floating value relative to other items in the mail list, this command returns the current position of a message whose information may have been retrieved during a prior POP3 session.

pop3_ID is a long integer reference to an open session created with POP3_Login.

uniqueID is a string value containing the Unique-ID of a message to locate on the POP3 server. This command will look for this value in the message headers of the account referenced by pop3_ID. Once found, the message's current position in the listing will be returned in msgNumber.

msgNumber is a long integer returned containing the current message number (its position within the current message list) of the item identified by uniqueID. If the uniqueID cannot be found on the server, a zero is returned in msgNumber and no error is returned.

4 IT Downloaded Mail

Downloading Mail, Overview

IT Downloaded Mail

version 6.5

The set of POP3 commands prefixed by "MSG_" allows the user to manipulate mail messages which have been saved as local files using the POP3_Download command described in the previous section. Because the set of commands are fully MIME compliant, 4D Internet Commands provide the means for attachments to be extracted and/or decoded. For more information on the MIME standards, refer to RFC#1521, RFC#1522 and RFC#2045.

Once messages have been downloaded to local files, the commands in this section provide a variety of functions to manipulate the documents. These commands can obtain information about the message parts, separate the header detail from the message body, detect and extract attachments within the message as well as delete existing documents.

MSG_FindHeader

version 6.5

MSG_FindHeader (fileName; headerLabel; headerValue) → Integer

Parameter	Type		Description
fileName	Text	→	Filename (path defaults to message folder)
headerLabel	String	→	Header label ("From:", "To:", "Subject:", etc.)
headerValue	Text	←	Value
Function result	Integer	←	Error Code

Description

Given a fileName of a message document retrieved to disk by the POP3_Download command, the command MSG_FindHeader will search the header section for headerLabel and return the value assigned to the field into headerValue.

fileName is the name of the file or the full path of the file of which to extract the header information. If only a filename is given, the path will default to the folder set by POP3_SetPrefs. If no folder has been specified by POP3_SetPrefs, the path will default to the folder containing the structure of the database.

headerLabel is a string containing the name of any header label. The headerLabel can reference any defined, user-defined or extended header such as "From:", "To:", "X-MyHeader", etc.

headerValue is a text variable or field where the command will return the value assigned to the specified header field.

See Also

POP3_Download, POP3_SetPrefs.

MSG_MessageSize

version 6.5

MSG_MessageSize (fileName; headerSize; bodySize; msgSize) → Integer

Parameter	Type		Description
fileName	Text	→	Filename (path defaults to message folder)
headerSize	Longint	←	Header size (subtracts linefeeds if Prefs ON)
bodySize	Longint	←	Body size (subtracts linefeeds if Prefs ON)
msgSize	Longint	←	Entire message or file size (ignores Prefs)
Function result	Integer	←	Error Code

Description

Given a fileName of a message document retrieved to disk by the POP3_Download command, the command MSG_MessageSize returns information about the sizes of the various portions of the message.

fileName is the name of the file or the full path of the file of which to return message information. If only a filename is given, the path will default to the folder set by POP3_SetPrefs. If no folder has been specified by POP3_SetPrefs, the path will default to the folder containing the structure of the database.

headerSize is the long integer variable returned containing the size of the header.

bodySize is the long integer variable returned containing the size of the body.

msgSize is the long integer variable returned containing the size of the message.

See Also

POP3_Download, POP3_SetPrefs.

version 6.5

MSG_GetHeaders (fileName; offset; length; headerText) → Integer

Parameter	Type		Description
fileName	Text	→	Filename (path defaults to message folder)
offset	Longint	→	Starting offset into headers (0 = start of header)
length	Longint	→	Number of characters
headerText	Text	←	Header text (removes linefeeds if Prefs ON)
Function result	Integer	←	Error Code

Description

The command MSG_GetHeaders returns the raw text of the entire header section of the message. The header portion of a POP3 message is defined as the text from the beginning of the message to the first occurrence of two consecutive carriage return/line feed sequences.

fileName is the name of the file or the full path of the file of which to extract the header. If only a fileName is given, the path will default to the folder set by POP3_SetPrefs. If no folder has been specified by POP3_SetPrefs, the path will default to the folder containing the structure of the database.

offset is the character position within the source header information at which to begin the retrieval.

length is the number of characters to return. The length of the header section can be determined with MSG_MessageSize.

headerText receives the text of the header.

See Also

MSG_MessageSize, POP3_SetPrefs.

MSG_GetBody

version 6.5

MSG_GetBody (fileName; offset; length; bodyText) → Longint

Parameter	Type		Description
fileName	Text	→	Filename (path defaults to message folder)
offset	Longint	→	Starting offset into message body (0 = start of body)
length	Longint	→	Number of characters
bodyText	Text	←	Body text (removes linefeeds if Prefs ON)
Function result	Longint	←	Error Code

Description

The command MSG_GetBody returns just the text of the message. It will not include enclosure text and will strip all MIME headers.

fileName is the name of the file or the full path of the file of which to extract the body of the message. If only a filename is given, the path will default to the folder set by POP3_SetPrefs. If no folder has been specified by POP3_SetPrefs, the path will default to the folder containing the structure of the database.

offset is the character position within the source body information at which to begin the retrieval.

length is the number of characters to return.

bodyText receives the text of the message body.

See Also

POP3_SetPrefs.

MSG_GetMessage

version 6.5

MSG_GetMessage (fileName; offset; length; rawText) → Integer

Parameter	Type		Description
fileName	Text	→	Filename (path defaults to message folder)
offset	Longint	→	Starting offset into message file (0 = start of file)
length	Longint	→	Number of characters
rawText	Text	←	Raw text (ignores Prefs)
Function result	Integer	←	Error Code

Description

The command MSG_GetMessage returns the raw text of the message regardless of enclosures. It does not strip MIME headers.

fileName is the name of the file or the full path of the file of which to extract the body of the message. If only a filename is given, the path will default to the folder set by POP3_SetPrefs. If no folder has been specified by POP3_SetPrefs, the path will default to the folder containing the structure of the database.

offset is the character position within the source message information at which to begin the retrieval..

length is the number of characters to return.

rawText receives the text of the entire message. The preference settings for linefeed stripping are ignored and no effort is taken to strip any attachment that may be embedded within the message body.

See Also

POP3_SetPrefs.

MSG_HasAttach

version 6.5

MSG_HasAttach (fileName; attachCount) → Integer

Parameter	Type		Description
fileName	Text	→	Filename (path defaults to message folder)
attachCount	Integer	←	Count of Attachments
Function result	Integer	←	Error Code

Description

If the file has attachments, the command MSG_HasAttach returns in the integer attachCount the number of attachments. An attachment is any non-text MIME enclosure. If the message has no attachments, 0 is returned.

fileName is the name of the file or the full path of the file of which to check for attachments. If only a filename is given, the path will default to the folder set by POP3_SetPrefs. If no folder has been specified by POP3_SetPrefs, the path will default to the folder containing the structure of the database.

attachCount is an integer value returned which specifies the number of attachments for fileName.

See Also

POP3_SetPrefs.

MSG_Extract

version 6.5

MSG_Extract (fileName; decode; attachmentPath; enclosureList) → Integer

Parameter	Type		Description
fileName	Text	→	Filename (path defaults to message folder)
decode	Integer	→	0 = No decoding, 1 = Decode if possible
attachmentPath	Text	→	FolderPath (path defaults to attachment folder)
enclosureList	Str | Txt Array	←	Enclosure filenames w/o FolderPath
Function result	Integer	←	Error Code

Description

The command MSG_Extract extracts all attachments and puts them into the attachments folder.

fileName is the name of the file or the full path of the file of which to extract the attachments. If only a filename is given, the path will default to the folder set by POP3_SetPrefs. If no folder has been specified by POP3_SetPrefs, the path will default to the folder containing the structure of the database.

decode is an integer specifying whether to attempt to decode the attachment. A value of zero indicates no attempt should be made to decode the attachment(s). A value of 1 will attempt to decode the file if it has been encoded in one of the following ways: Binhex, AppleSingle, AppleDouble, or Base64.

attachmentPath is the FolderPath of where to save the attachment. If no FolderPath is specified, the file will be saved in the attachments folder as specified in POP3_SetPrefs. If no FolderPath has been specified with POP3_SetPrefs, the attachment will be saved in the same folder as the database structure.

enclosureList is a text/string array which is returned containing the file names of each attachment. Only the document name will be returned in each array element, not the full pathname. If a string array is used, the array should be sized to at least 31 characters, since the maximum length of a document name is 31 characters.

See Also

POP3_SetPrefs.

MSG_Delete

MSG_Delete (fileName; folder) → Integer

Parameter	Type		Description
fileName	Text	→	Filename (path defaults to message folder)
folder	Integer	→	0 = Message Folder, 1 = Attachment Folder
Function result	Integer	←	Error Code

Description

The command MSG_Delete deletes a local file.

fileName is the name of the file or the full path of the file to delete. If only a filename is given, the path will default to the folder based on the following: If folder is not null (0 or 1), the path will use that folder as specified by POP3_SetPrefs. If no folder has been specified by POP3_SetPrefs for that folder kind (message or attachment), the path will default to the folder containing the structure of the database. If folder is null, the path will default to the folder containing the structure of the database.

folder is an integer value which can be specified when fileName does not indicate the full pathname to the document. The folder parameter specifies which folder the document named fileName resides in. A zero value indicates the file resides in the message folder. A value of 1 indicates the file resides in the attachments folder.

Warning: This command will delete **ANY** file passed to it. Be very careful when using this command.

See Also

POP3_SetPrefs.

5 IT File Transfer

File Transfer, Overview

IT File Transfer

version 6.5

The File Transfer Protocol (FTP) is the primary means of transferring documents and applications from one computer to another. FTP "sites" are computers throughout the world running FTP server software. The File Transfer Protocol provides a means for disparate systems to exchange files. Client applications on a variety of platforms can all log into a FTP server in order to upload or download text or binary files. The FTP routines within the 4D Internet Commands give developers the tools to create FTP clients within their 4D databases.

Note: When specifying pathnames in the FTP commands, you should always treat file locations on the FTP site as a Unix directory, even if you know the FTP host to be a Macintosh running FTP server software. Whatever the platform, the FTP server software will internally convert your unix pathname to the format it needs to serve its documents to connected clients.

FTP_Progress

version 6.5

FTP_Progress (left; top; windowTitle; thermoText; cancel) → Integer

Parameter	Type		Description
left	Integer	→	Left window coordinate
top	Integer	→	Top window coordinate
windowTitle	String	→	Thermometer Window Title
thermoText	String	→	Text above thermometer progress
cancel	String	→	Cancel button text
Function result	Integer	←	Error Code

Description

The command FTP_Progress defines window coordinates and dialog box text for the FTP progress indicator. The progress indicator can appear during calls to either FTP_Send or FTP_Receive. The FTP_Progress command does not display the progress window itself, it only defines the windows characteristics for when it is displayed by the send and receive commands. Both FTP_Send and FTP_Receive have parameters which can show or hide the progress window.

The progress window will automatically close upon completion of a file transfer. If for some reason the size of the file being sent or received cannot be determined, the thermometer will be displayed as a barber pole and the file size will be displayed as "unknown".

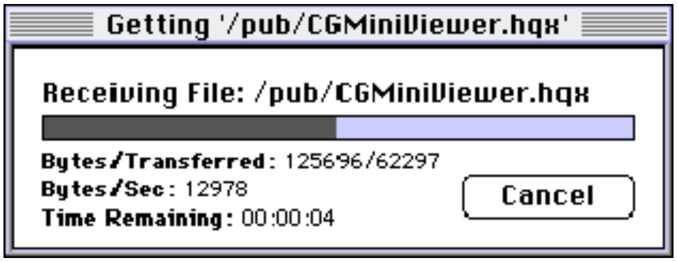
left is the coordinates of the left side of the thermometer progress window. If left is -1, the window will be centered horizontally on the screen.

top is the coordinates of the top side of the thermometer progress window. If top is -1, the window will be centered vertically on the screen.

windowTitle is the title of the thermometer progress window. In the following example, the window title is "Getting '/pub/CGMiniViewer.hqx'" If windowTitle is a null string, the window will have no title.

thermoText is the text to be displayed above the progress thermometer. If thermoText is "*" then the text will be the default. In the following example, thermoText is "Receiving File: /pub/CGMiniViewer.hqx". The default text for the thermometer is the status text of the transfer process, sent by the host. This text changes as the connection goes through the different stages of the transfer process.

cancel is the text of the **Cancel** button. If cancel is a null string, the **Cancel** button will be hidden. If cancel is "*", the text will be the default text, which is "Cancel".


Getting '/pub/CGMiniViewer.hqx'

Receiving File: /pub/CGMiniViewer.hqx

Bytes/Transferred: 125696/62297
Bytes/Sec: 12978
Time Remaining: 00:00:04

Example

```
⇒     $error:=FTP_Progress (-1;-1;"FTP File Transfer";"*";"*")
      Case of
         : (FTP_Login ("ftp.acius.com";"anonymous";vEmailID;vFTP_ID;vFTP_Msg)#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   `all commands executed without error
      End case
```

version 6.5

FTP_Login (hostName; userName; password; ftp_ID{; welcomeText}) → Integer

Parameter	Type		Description
hostName	String	→	Host name or IP address
userName	String	→	User name
password	String	→	Password
ftp_ID	Longint	←	Reference to this new FTP session
welcomeText	Text	←	FTP Welcome text
Function result	Integer	←	Error Code

Description

The command FTP_Login establishes a connection with the FTP server at hostName and logs onto the system using the supplied userName and Password.

hostName is the host name or IP address of the remote system.

userName is the name of a user account recognized by the FTP server. Many FTP servers support guest access via an "anonymous" username. If you are logging in anonymously, it is customary to supply your e-mail address as the password.

password is the password for userName on the system.

ftp_ID is the long integer value obtained for the newly opened session. This value will be used in subsequent FTP commands. This parameter must be passed a 4D variable or field in order to accept the returned results.

welcomeText is an optional parameter which contains the text returned when the user logs into the system. Many FTP sites have a Welcome message displayed at the time of login. If specified, this parameter must be passed a 4D variable or field in order to accept the returned results.

Example

```
        $OK:=False
        Case of
⇒           : (FTP_Login ("ftp.acius.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   `all commands executed without error
        End case
```

See Also

FTP_Logout.

FTP_GetDirList

FTP_GetDirList (ftp_ID; directory; names; sizes; kinds; modDates) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
directory	Text	→	Unix directory pathname
		←	Current directory
names	Str | Txt Array	←	Listing Names
sizes	Longint Array	←	Listing Sizes
kinds	Integer Array	←	Listing Kinds 0 = plain file, 1 = directory, 2 = block-type special file, 3 = character-type special file, 4 = symbolic link, 5 = FIFO special file, 6 = AF_UNIX address family socket)
modDates	Date Array	←	Listing Modification Dates
Function result	Integer	←	Error Code

Description

The command FTP_GetDirList will retrieve a listing of the objects in a directory of the FTP session identified by ftp_ID. Information on the names, sizes, types and modification dates of the directory items is returned into four arrays. A connection to the FTP site must have already been opened via FTP_Login and still valid (FTP_VerifyID). The FTP_GetDirList command changes your current working directory (CWD) to the path given and returned to the directory parameter.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

directory is a text value in the format of a *HostPath* which references a FTP directory. A 4th Dimension variable or field must be passed to this parameter since the resulting "current directory" will be returned. Normally, the value returned to this parameter will be the same as the value passed to it. However, there may be cases (such as access restrictions) where the directory change was not successful. In this case, the directory parameter will hold the HostPath to the session's current directory.

A null string passed in this parameter will return the current directory file listing into the arrays and the current directory's HostPath into the directory parameter.

names is a string or text array to hold the name of each object in the specified directory.

sizes is a long integer array to hold the sizes of the objects in directory.

kinds is an integer array to hold the type of each object in directory. The interpretation of this value is based on the following table:

Kind	**File Type**
0	plain file
1	directory
2	block-type special file
3	character-type special file
4	symbolic link
5	FIFO special file
6	AF_UNIX address family socket

modDates is a 4D date array to hold the last modified date for each object in directory.

See Also

FTP_Login, FTP_VerifyID.

FTP_GetFileInfo (ftp_ID; hostPath; size; modDate) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
hostPath	Text	→	Pathname to document
size	Longint	←	Size of Document
modDate	Date	←	Modification Date
Function result	Integer	←	Error Code

Description

Given a pathname to a file in the format of a *HostPath*, the command FTP_GetFileInfo returns the size and last modification date of the file.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

hostPath is the text path to the document to return information about.

size is a long integer variable or field to hold the size of the file identified by hostPath.

modDate is a date variable or field to hold the last modification date of the file.

See Also

FTP_GetDirList.

FTP_VerifyID

FTP_VerifyID (ftp_ID) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
		←	0 = Connection has already closed
Function result	Integer	←	Error Code

Description

A FTP server will disconnect accounts which do not show activity in a period of time determined by its administrator. Each command that interacts with the FTP server will force a reset of your inactivity timer. The FTP_VerifyID command resets the inactivity time for the specified FTP session without altering the current state or directory. This allows the user to keep a session active if the possibility exists that the session may timeout.

When executed, the FTP_VerifyID command will verify that the connection has not already been closed. If the session is still open the command will tell the FTP server to reset the timeout counter for the session back to zero. If the connection has already closed, FTP_VerifyID will return the appropriate error, free memory used by the FTP session, and return a zero value back to ftp_ID.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

See Also

FTP_Login.

version 6.5

FTP_MakeDir (ftp_ID; directory) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
directory	Text	→	Unix directory pathname
Function result	Integer	←	Error Code

Description

Given an acceptable directory name, the command FTP_MakeDir will create a new Folder directory within the Current Working Directory (CWD). Calls to the FTP_GetDirList command will change your CWD to the specified directory. An error will be returned if you do not have sufficient access priveleges to perform this action.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

directory is a text value in the format of a *HostPath* which references a FTP directory. The value of the directory parameter may be a full pathname specification or simple folder name. If the shortened form is used then the directory is created within the CWD. It is recommended that the directory name **not** contain any blank spaces.

See Also

FTP_GetDirList, FTP_RemoveDir.

FTP_RemoveDir

version 6.5

FTP_RemoveDir (ftp_ID; directory) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
directory	Text	→	Unix directory pathname
Function result	Integer	←	Error Code

Description

Given an acceptable directory name, the command FTP_RemoveDir will delete a Folder directory. Calls to the FTP_GetDirList command will change your CWD to the specified directory. An error will be returned if you do not have sufficient access privileges to perform this action. Additionally, attempting to remove a directory which contains items will likely result in a security error.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

directory is a text value in the format of a *HostPath* which references an existing FTP directory. The value of the directory parameter may be a full pathname specification or simple folder name. If the shortened form is used then the specified directory must be within the CWD.

See Also

FTP_GetDirList, FTP_MakeDir.

FTP_Rename

version 6.5

FTP_Rename (ftp_ID; hostPath; newPathName) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
hostPath	Text	→	Pathname to document on FTP Server
newPathName	Text	→	New document name
Function result	Integer	←	Error Code

Description

Given a pathname to a file in the format of a hostPath, the command FTP_Rename will rename the specified file on the remote FTP Server. An error will be returned if you do not have sufficient access priveleges to perform this action.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

hostPath is the text path to the document to be renamed. The value of the hostPath parameter may be a full pathname specification or simple file name. If the shortened form is used then the specified file must be within the CWD.

newPathName contains the value you wish to rename the remote document.

FTP_Delete

version 6.5

FTP_Delete (ftp_ID; hostPath) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
hostPath	Text	→	Pathname to document
Function result	Integer	←	Error Code

Description

Given a pathname to a file in the format of a *HostPath*, the command FTP_Delete will delete the specified file from the remote FTP Server. An error will be returned if you do not have sufficient access priveleges to perform this action.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

hostPath is the text path to the document to be deleted. The value of the hostPath parameter may be a full pathname specification or simple file name. If the shortened form is used then the specified file must be within the CWD.

See Also

FTP_RemoveDir.

FTP_MacBinary

FTP_MacBinary (ftp_ID; macBinaryMode) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
macBinaryMode	Integer	→	-1 = Get Current setting, 1 = Enable, 0 = Disable
		←	Current setting (if -1 passed)
Function result	Integer	←	Error Code

Description

The command FTP_MacBinary enables/disables the MacBinary mode setting for FTP transfers using FTP_Send and FTP_Receive. Given a current FTP session identified by ftp_ID, this command will either turn MacBinary transfers on or off depending on the value passed in the macBinaryMode parameter.

The MacBinary protocol is often used by Macintosh FTP clients and servers to facilitate the transfer of binary data or files that contain both data and resource forks.

Note for Windows users: It is possible to use the MacBinary protocol for FTP transfers in a Windows environment however it should be noted that it may often not make sense to decode a MacBinary file on a PC computer. Intel-based machines cannot store files containing both data and resource forks. Since such a file format is foreign to the PC platform, Macintosh files which contain a resource fork are likely to be corrupted if saved in an unencoded format.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

macBinaryMode is integer parameter indicating whether to turn MacBinary transfers on or off. This value should be passed as a variable so the command can return the state of MacBinary transfers after the attempted change. Passing a 1 will enable MacBinary and a zero will disable. A -1 value in the parameter will cause the command to return in the macBinaryMode parameter the current state of MacBinary transfers (1 or zero). Not all FTP servers support the MacBinary protocol, so it is possible that you may attempt to enable it but the command returns back a zero indicating the MacBinary was not enabled.

Example

This example enables the MacBinary protocol before receiving an FTP file. If the file was successfully received with MacBinary turned on then it is decoded into its original format and the MacBinary document is deleted.

```
     vLocalFile:=""
     vUseMacBin:=1
        `Try to turn MacBinary on for the download
⇒    $error:=FTP_MacBinary (vFTP_ID;vUseMacBin)
     $error:=FTP_Receive (vFTP_ID;"MyApplication";vLocalFile;cbShowTherm)
     If ($error=0) & (vUseMacBin=1)  `If received OK and the file is in MacBinary format
        vDecodePath:=""
        If (IT_Decode (vLocalFile;vDecodePath;8)=0)  `MacBinary decode
        DELETE DOCUMENT(vLocalFile)  `If sucessful decode of source, then delete it.
        End if
     End if
```

See Also

IT_Decode.

FTP_Send

FTP_Send (ftp_ID; localPath; hostPath; progress) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
localPath	Text	→	Pathname of document to send
hostPath	Text	→	Pathname to destination of document
progress	Integer	→	1 = Show Progress, 0 = Hide Progress
Function result	Integer	←	Error Code

Description

Given a reference to an open FTP session, the pathname of a document to send and the destination pathname, the command FTP_Send sends the document to the remote machine. FTP_Send will return immediately if a FTP file status error occurs.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

localPath is the path of the document to be sent. If localPath is a null string, the user will be presented with the Standard Open File dialog. If localPath is a file name with no path, the command will look in the folder that contains the database structure for the file. As with all paths to local documents, the directories should be seperated by a delimiter appropriate for the platform. For more information, see the section entitled Glossary and Terminology at the beginning of the manual.

hostPath is the path to the destination of the document, including the file name. The hostPath represents the desired name of the file once it has been received by the FTP server. If localPath is a null string allowing the user to pick a file from disk, then hostPath may also be a null string, in which case the chosen file's name will be used.

hostPath may be either a full pathname specification or simply a filename. If a full pathname is supplied, the specified file will be placed in the directory indicated by hostPath. If only a filename is provided, or null strings are used in the file selection, then the file will be sent to the last directory navigated to by the FTP_GetDirList command (the Current Working Directory [CWD]).

If the file or pathname cannot be resolved correctly, the command will return an error. If the user does not have enough privileges to send a file to that directory, an error will be returned. As with all paths to Unix documents, the path should be separated by slashes ("/"). For more information, see the section entitled Glossary and Terminology at the beginning of the manual.

progress is an integer value indicating whether the Progress indicator should be displayed or not. A value of 1 will display the progress indicator. A value of zero will hide the progress indicator.

Examples

Example 1

```
        $OK:=False
        Case of
           : (FTP_Login ("ftp.acius.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   `all commands executed without error
        End case
```

Example 2

```
⇒       $error:=FTP_Send (vFTP_ID;"";"";1)
```

See Also

FTP_Progress, FTP_Receive.

FTP_Append

version 6.5

FTP_Append (ftp_ID; localPath; hostPath; progress) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
localPath	Text	→	Pathname of document to send
hostPath	Text	→	Pathname to destination of document
progress	Integer	→	1 = Show Progress, 0 = Hide Progress
Function result	Integer	←	Error Code

Description

The command FTP_Append performs the same action as FTP_Send with the one exception that it will append the data being sent to the end of an existing file identified by the hostPath parameter. This command's primary function is to append data onto the end of pre-existing text files.

See Also

FTP_Send.

FTP_GetType (ftp_ID; ftp_Mode) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
ftp_Mode	String	←	"A" = Ascii; "I" = Image; "L 8" = Logical 8-bit
Function result	Integer	←	Error Code

Description

The command FTP_GetType returns information about the current FTP Transfer mode. The Transfer mode may be set using the FTP_SetType command.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

ftp_Mode returns a code describing the current FTP transfer mode.

See Also

FTP_SetType.

FTP_SetType

FTP_SetType (ftp_ID; ftp_Mode) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
ftp_Mode	String	→	"A" = Ascii; "I" = Image; "L 8" = Logical 8-bit
Function result	Integer	←	Error Code

Description

The command FTP_SetType is used to alter the FTP transfer mode used during Send/Receive operations. Typically this will not need to be changed from the default settings. However, because of differences between various platforms and ftp implementations, it may be necessary to change the mode for certain types of FTP transfers. In particular, some transfers of plain-text documents may require you to switch the mode to Ascii in order to properly transfer the text file.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

ftp_Mode should contain a code as described above indicating the preferred transfer mode to use for future Send/Receive operations.

See Also

FTP_GetType.

FTP_System

FTP_System (ftp_ID; systemInfo) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
systemInfo	String	←	System Information
Function result	Integer	←	Error Code

Description

The command FTP_System simply obtains information into systemInfo that describes the FTP server software.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

systemInfo will contain information about the FTP server.

FTP_Receive

FTP_Receive (ftp_ID; hostPath; localPath; progress) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
hostPath	Text	→	Pathname of document to receive
localPath	Text	→	Pathname to destination of document
		←	Resulting file pathname (if "" passed)
progress	Integer	→	0 = Hide Progress, 1 = Show Progress
Function result	Integer	←	Error Code

Description

The command FTP_Receive receives a file using the File Transfer Protocol from the path referenced by hostPath. FTP_Receive will return an error# -48 if the destination file already exists.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

hostPath is a text value that specifies the path of the document to be received. If hostPath is not a full path to a document, the command will return an error. As with all paths to Unix documents, the path should be separated by slashes ("/"). For more information, see the section entitled Glossary and Terminology at the beginning of the manual.

localPath is a text value that specifies the path of the destination of the document. If localPath is a null string, the user will be presented with a Standard Save-File Dialog and the resulting file pathname will be returned back into the localPath variable. If localPath contains only a filename, the file will be saved in the same folder as the structure of the database. As with all paths to local documents, the path should be separated by the delimiter appropriate for the platform the externals are being used. For more information, see the section entitled Glossary and Terminology at the beginning of the manual.

progress is an integer value indicating whether the Progress indicator should be displayed or not. A value of 1 will display the progress indicator. A value of zero will hide the progress indicator.

Example

```
     vLocalFile:=""
     vUseMacBin:=1
        `Try to turn MacBinary on for the download
     $error:=FTP_MacBinary (vFTP_ID;vUseMacBin)
⇒    $error:=FTP_Receive (vFTP_ID;"CGMiniViewer.hqx";vLocalFile;cbShowTherm)
     If ($error=0) & (vUseMacBin=1)
        vDecodePath:=""
        If (IT_Decode (vLocalFile;vDecodePath;8)=0)  `MacBinary decode
           DELETE DOCUMENT(vLocalFile)  `If successful decode of source, then delete it.
        End if
     End if
```

See Also

FTP_MacBinary, IT_Decode.

FTP_Logout

version 6.5

FTP_Logout (ftp_ID) → Integer

Parameter	Type		Description
ftp_ID	Longint	→	Reference to a FTP login
		←	0 = session successfully closed
Function result	Integer	←	Error Code

Description

Given a reference to an open FTP session, the command FTP_Logout will disconnect from the server and free any memory used by the session. This command will return the value of zero into the ftp_ID parameter upon successful close of the session.

ftp_ID is the long integer reference to the FTP session established with FTP_Login.

Example

```
If (FTP_Login ("ftp.acius.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
```

See Also

FTP_Login.

6 IT TCP/IP

Low Level Routines, Overview

IT TCP/IP

version 6.5

TCP/IP or Transmission Control Protocol/Internet Protocol, is the primary protocol used for sending data over the internet. The TCP commands included with 4D Internet Commands allow developers to establish TCP session and send and receive TCP packets via these sessions.

There are two ways to establish a TCP connection. The first way is to execute the TCP_Open command. This will open a connection with the domain specified on the specified port. The other way to open a connection is to execute the TCP_Listen command. This command will open a connection with the specified domain on the specified port, and will listen for an incoming connection. The best way to determine if a connection has been established is to check the state of the session with the command TCP_State upon completion of the TCP_Listen command. A status code will be returned which will correspond to the current state of the session. From here you can send and/or receive TCP packets as you could with a connection established with TCP_Open.

The low-level TCP/IP commands require advanced knowledge about the protocols of communication. Developers using these routines should have a complete understanding of any protocol they attempt to implement. Information about the various TCP/IP assigned port numbers, communication protocols, addressing requirements, etc. can be found in the RFCs.

TCP_Open

TCP_Open (hostName; remotePort; tcp_ID) → Integer

Parameter	Type		Description
hostName	String	→	Host name or IP address
remotePort	Integer	→	The remote port to connect to (0 = any)
tcp_ID	Longint	←	Reference to this TCP session
Function result	Integer	←	Error Code

Description

The command TCP_Open initiates an outgoing TCP connection to a domain.

TCP_Open initiates a connection to the remote TCP referenced by hostName, on the port referenced by remotePort (if not 0). A long integer value will be returned to tcp_ID, which will be used by all subsequent TCP calls referring to the session. TCP_Open is set to time out in 30 seconds if no data is received by the session identified by the tcp_ID parameter. The default timeout value can be changed for all commands via IT_TimeOut.

hostName is the host name or IP address of the machine that you are opening a connection to.

remotePort indicates the TCP port on the machine indicated by hostName that with which you wish to establish a connection.

tcp_ID is the long integer reference to the session that was opened. This reference will be used in all subsequent TCP external calls that reference this session.

See Also

IT_TimeOut.

TCP_Listen

TCP_Listen (remoteHost; remotePort; localPort; timeout; tcp_ID) → Integer

Parameter	Type		Description
remoteHost	String	→	Host name or IP address
remotePort	Integer	→	Port number to be listening
localPort	Integer	→	Local port number, 0 = find an unused port to use
		←	Used local port number (if 0 passed)
timeout	Integer	→	# of seconds to wait, 0 = wait forever
tcp_ID	Longint	←	Reference to this TCP session
Function result	Integer	←	Error Code

Description

The command TCP_Listen waits for a connection to be made from the machine referenced by remoteHost on the port referenced by remotePort. This command does not return control back to the 4th Dimension calling method until either a connection is made or the timeout period has elapsed. Though it may seem as though this would lock up your database until a connection was made, the command is friendly to other 4th Dimension processes that may be running. This command will slice time to other 4D processes you may already have running.

Most developers will want to issue this call from a method which has been spawned into its own 4D process (especially if you specify the timeout period to wait forever).

remoteHost is the host name or IP address of the machine that you are waiting for a connection from. If a null string is passed in this parameter, this command will accept an incoming connection from any machine.

remotePort identifies the port number to be listening for an incoming connection.

localPort contains the port on your local machine you wish to use for communication. If you pass a zero as this parameter, the command will find any unused port and pass that number back to this parameter.

timeout specifies the number of seconds this command will wait for an incoming connection. A zero in this parameter will cause the command to wait indefinitely for an incoming connection. Caution should be taken when passing a zero since control will never be returned to the calling 4D process if a connection is never made. Never pass zero to this parameter in a single-process database.

tcp_ID is the long integer reference to the session that was opened. This reference will be used in all subsequent TCP external calls that reference this session.

Example

```
        C_LONGINT(vTCPID)
        C_INTEGER(vStatus)
⇒       $err:=TCP_Listen ("";49152;0;30;vTCPID)
        $err:=TCP_State (vTCPID;vStatus)
        If (vStatus=8)      `connection was established
            DoSomething
            $err:=TCP_Close (vTCPID)
        End if
```

See Also

Appendix B, TCP Port Numbers, TCP_Open, TCP_State.

TCP_Send

version 6.5

TCP_Send (tcp_ID; sendText) → Longint

Parameter	Type		Description
tcp_ID	Longint	→	Reference to an open TCP session
sendText	Text	→	Text to send
Function result	Longint	←	Error Code

Description

The command TCP_Send sends data to the TCP session designated by tcp_ID.

tcp_ID is a long integer reference to an open TCP session as established with either the TCP_Open or TCP_Listen command.

sendText is a text value to be sent to the TCP session referenced by tcp_ID.

See Also

TCP_Listen, TCP_Open.

TCP_Receive

TCP_Receive (tcp_ID; text) → Integer

Parameter	Type		Description
tcp_ID	Longint	→	Reference to an open TCP session
text	Text	←	Received Text
Function result	Integer	←	Error Code

Description

Given a long integer reference to an established TCP Session, the command TCP_Receive receives packets of data into text.

tcp_ID is a long integer reference to an open TCP session as established with either the TCP_Open or TCP_Listen command.

text is the text received. When receiving data via TCP packets, you cannot count on all of your data being received by a single TCP_Receive call. The TCP_Receive command is usually called within a Repeat loop which continually checks on the status of the connection or is scanning for a known value.

Example

```
      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))  until host closes connection or an error
```

See Also

TCP_Send.

TCP_State

version 6.5

TCP_State (tcp_ID; statusCode) → Integer

Parameter	Type		Description
tcp_ID	Longint	→	Reference to an open TCP
statusCode	Integer	←	TCP status code
Function result	Integer	←	Error Code

Description

The command TCP_State returns the state of a particular TCP connection.

tcp_ID is a long integer reference to an open TCP session as established with either the TCP_Open or TCP_Listen command.

statusCode is an integer variable returned which corresponds to one of the status codes below.

0	Connection Closed
2	Listening for an incoming connection
8	Established

Example

This example assumes that a valid TCP connection has already been established and is identified by the Longint value assigned to the *$tcp_id* variable. In this example, a command is sent to a web server requesting a page of information and then a loop is entered to receive the results. Since web servers automatically close their connnections once they have performed their action, this example keeps receiving until the connection is dropped or an error occurs.

```
      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
```

See Also

TCP_Listen, TCP_Open.

TCP_Close

version 6.5

TCP_Close (tcp_ID) → Integer

Parameter	Type		Description
tcp_ID	Longint	→	Reference to an open TCP session
		←	0 = Session successfully closed
Function result	Integer	←	Error Code

Description

The command TCP_Close closes the TCP session referenced by tcp_ID. If a TCP session is not closed, it will occupy one of the 64 references available for TCP sessions. If there are 64 sessions open which have not been closed, the user will not be able to open another session.

tcp_ID is a long integer reference to an open TCP session as established with either the TCP_Open or TCP_Listen command. This command will return the value of zero into the tcp_ID parameter upon successful close of the session.

See Also

TCP_Listen, TCP_Open.

7 IT Internet

Special Internet Commands, Overview

IT Internet

version 6.5

The set of commands included in this section can be used to perform common tasks over the Internet. Included in this section are commands to 'Ping' and 'Finger' a machine, obtain the time from a time server, resolve a domain name or IP address, and convert a domain name or IP address from or to a long integer. These commands are often used in conjunction with the other 4D Internet Commands.

NET_Finger

version 6.5

NET_Finger (hostName; searchText; results) → Integer

Parameter	Type		Description
hostName	String	→	Host name or IP address
searchText	String	→	Search text
results	Text	←	Finger results
Function result	Integer	←	Error Code

Description

Given an IP address of a machine to search and the text of a user account name on the machine, the command NET_Finger will return the finger results into results. The unix finger command is designed to return information about the last login time of a user as well as any additional information the user chooses to provide within their ".plan" and ".project" files.

Two different routes may be specified for the Finger search. A finger search may be attempted directly at the user's machine. For instance, to get information about "johnt" at "acius.com", you could perform the search as:

```
⇒	$error:=NET_Finger ("www.acius.com";"johnt";$fingertext)
```

The same finger search could also be performed indirectly. An indirect search would ask a remote server which supports the finger command to perform your query. For instance, the following will ask the machine identified by the domain name "acius.com" to perform a remote finger query of the user "johnt@acius.com".

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

While the main information returned in each case should be the same, there are likely to be some subtle differences in the returned text. Different machines may have different options configured when they execute the finger command and the results could vary slightly. Also, there is likely to be some formatting difference between the results of a direct and indirect finger command, with indirect searches often containing additional linefeeds.

hostName is the host name or IP address of the machine in which the user identified by searchText has an account.

searchText is either the text to search for on the given finger server, or a machine name or IP address. If searchText is a user name, the command will search through the directory of user names on that server for searchText. If searchText is a machine name or IP address, the command will send a finger request through the finger server in hostName to the machine specified.

results is the text returned which contains the results of the search.

NET_Ping

IT Internet

version 6.5

NET_Ping (hostName; text; alive{; timeout}) → Integer

Parameter	Type		Description
hostName	String	→	Host name or IP address
text	Text	→	Text to send in ping
alive	Integer	←	1 = Alive, 0 = Timeout/Inactive
timeout value	Integer	→	# of seconds to wait, 0 = use IT_SetTimeOut
Function result	Integer	←	Error Code

Description

The command NET_Ping provides a mechanism to query a remote IP address to see if it is currently active. If the pinged machine is currently running the TCP/IP protocol and the network between the two sites is functional, an 'Alive' status should be returned. Typically, the pinged machine provides no indication to its user of the activity, assuming that the machine can respond to ping (ICMP Echo) requests.

NET_Ping will ping a machine specified by either a host name or IP address. Any machine with an IP address that is accessible via the network can be pinged. This includes end-user machines. [Some security systems known as "firewalls" may hinder you from pinging machines under their protection.]

hostName is the host name or IP address of the machine to ping.

text is the text to send in the ping. The text parameter exists only to effect the size of the TCP packet being sent as the Ping command is executed.

alive is the integer returned corresponding to the state of the machine pinged. A value of 1 indicates the machine is alive. A value of zero indicates the machine is either inactive or the ping timed out before a response was received.

timeout specifies the number of seconds this command will wait for the Ping to complete. This is an optional parameter which, if not supplied, will default to zero. A zero value in this parameter will cause the command timeout if a response is not received by the number of seconds specified

See Also

IT_SetTimeOut.

NET_Time

IT Internet

version 6.5

NET_Time (hostName; date; time; offset) → Integer

Parameter	Type		Description
hostName	String	→	Host name or IP address
date	Date	←	Date
time	Longint	←	Time, expressed as seconds since midnight
offset	Integer	→	Hours to offset
Function result	Integer	←	Error Code

Description

Given the host name or IP address to an Internet Time server, the command NET_Time will obtain the current date and time from the machine and apply any offset needed to convert to the user's local time.

Note: This command does not affect the computer's internal clock.

hostName is the host name or IP address of an Internet Time server.

date is a 4D Date returned, containing the resulting date after the offset has been applied.

time is a LongInt value returned, containing the resulting time after the offset has been applied. The value represents the seconds since midnight on date. See the example below for a method to convert this value to a 4D Time variable.

offset is the number of hours to add or subtract from the base time values. Internet Time Servers express their values in Universal Time (Greenwich Mean Time). Even if the time server is in your geographic region, it is likely that you will need to supply an offset value to compensate for the difference between your local time and Universal time.

Example

The following example obtains the Universal Time from the Time server at "apple.com". The command then subtracts the seven hours specified as the Offset and returns the resulting Date and Time (Time is expressed as a Longint value, which can then be converted using 4D's Time string command, as below).

```
	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))  `Convert the LongInt time to a 4D Time
	End if
```

See Also

Time string.

NET_NameToAddr

NET_NameToAddr (hostName; ip_Longint) → Integer

Parameter	Type		Description
hostName	String	→	Host name or IP address
ip_Longint	Longint	←	Long Integer reference to the address
Function result	Integer	←	Error Code

Description

The command NET_NameToAddr takes a host name or IP address and returns a unique long integer reference to the address.

hostName is the host name or IP address.

ip_Longint is a Longint value representing the IP address specified in the hostName parameter. All IP address strings can be converted to a signed Longint value.

While the ip_Longint value does not typically have a significant use, some developers may find this command useful to convert IP addresses into a more compact Longint format for data-storage.

See Also

NET_AddrToName.

NET_AddrToName

version 6.5

NET_AddrToName (ip_Longint; hostName; ip_Address) → Integer

Parameter	Type		Description
ip_Longint	Longint	→	Long Integer reference to the address
hostName	String	←	Host name
ip_Address	String	←	IP address
Function result	Integer	←	Error Code

Description

The command NET_AddrToName takes a long integer reference to a host name, and returns both the host name and the IP address of that host.

ip_Longint is the long integer reference to an IP address.

hostName is the string returned which contains the host name. If the host name is not able to be resolved, ip_Address will return a null string and no error will be returned.

ip_Address is the string returned which contains the IP address.

See Also

NET_NameToAddr.

NET_Resolve

NET_Resolve (hostName; ipOrHost) → Integer

Parameter	Type		Description
hostName	String	→	Host name or IP address
ipOrHost	String	←	Returns the opposite value
Function result	Integer	←	Error Code

Description

Given a host name in the first parameter, the command NET_Resolve will return the IP address into the second parameter. If the first parameter is passed an IP address, the second parameter will yield the registered host name for that machine.

hostName is a string which contains either an IP address or a host name.

ipOrHost - If the first parameter contained a Host Name then this parameter will receive its IP address. If the IP address was specified in the first parameter then this value will receive its Host Name.

Example

The following example first passes a host name "www.netcom.com" to the NET_Resolve command in order to obtain its IP address. The example then makes another call to the command, passing it the IP address in order to obtain its registered host name.

```
    C_BOOLEAN($ERR)
    C_STRING(80;$Resolved)  `Can be any sized string or text value
⇒   $ERR:=ERRCHECK ("NET_Resolve";NET_Resolve ("www.netcom.com";$Resolved))
        `$Resolved was returned the value '192.100.81.100'
⇒   $ERR:=ERRCHECK ("NET_Resolve";NET_Resolve ($Resolved;$Resolved))
        `$Resolved was returned the value 'www.netcom.com'
```

8 IT Utilities

Utility Commands, Overview

IT Utilities

version 6.5

The commands within this section provide various utilities which are supportive of the other sections of 4D Internet Commands. Many of these commands help the developer determine the environment in which a user's machine is operating, the versions of the software and the state and IP address of their computer.
Other commands within this section help the developer decipher error codes, encode and decode files, and effect the default timeout value for many of the commands in all sections.

IT_MacTCPInit

IT_MacTCPInit → Integer

Parameter	Type		Description
This command does not require any parameters			
Function result	Integer	←	Error Code

Description

The command IT_MacTCPInit opens the TCP driver for use with the 4D Internet Commands. The command acts as a function and returns an integer value error if the TCP driver cannot be opened.

It is recommended that users place this command in the On Startup Database Method of their 4th Dimension database. The command only needs to be executed once. Users of dial-up schemes which are utilized by TCP such as PPP or SLIP may wish to delay use of this command until a TCP connection is needed as use of this command will cause these commands to open and initialize their dial-up connection.

Note: This command must be executed before any 4D Internet command can be executed.

IT_Platform

IT_Platform → Integer

Parameter	Type		Description
This command does not require any parameters			
Function result	Integer	←	Platform Type (0 = 68K Code, 1 = PPC Code, 2 = Windows)

Description

The function IT_Platform returns an integer value indicating which set of 4D Internet Commands code is currently executing. The function will return a zero if running the 68K code, a one if running the PPC native code or a 2 if running on Windows.

Example

```
      C_BOOLEAN (<>ITnative)
⇒     <>ITnative:=(IT_Platform=1)
```

IT_Version

IT_Version → String

Parameter	Type		Description
This command does not require any parameters			
Function result	String	←	Version String

Description

The function IT_Version returns a String value indicating the version number of 4D Internet Commands.

Example

The following example presents an alert dialog to the user indicating what version of 4D Internet Commands they are using.

```
⇒    ALERT("4D Internet Commands version: "+IT_Version)
```

IT_TCPversion

version 6.5

IT_TCPversion (stackKind; stackVersion) → Integer

Parameter	Type		Description
stackKind	Integer	←	0 = None, 1 = MacTCP, 2 = Open Transport, 3 = WinSock
stackVersion	Text	←	Version number of the TCP stack
Function result	Integer	←	Error Code

Description

The command IT_TCPversion returns information about the type of TCP stack currently in use by the 4D Internet Commands. The type of Stack varies by platform. On the Macintosh, both MacTCP and Open Transport are supported. Under Windows, the WinSock TCP stack is supported.

stackKind returns an integer value expressing the type of TCP stack currently in use. The value returned identifies the following supported TCP stacks:

Code	TCP Stack
0	None
1	MacTCP
2	Open Transport
3	WinSock

stackVersion returns a Text value representing the version number of the TCP stack currently in use and identified by the stackKind parameter.

IT_MacTCPVer

version 6.5

IT_MacTCPVer (versionCode) → Integer

Parameter	Type		Description
versionCode	Integer	←	Version Code of MacTCP installed
Function result	Integer	←	Error Code

Description

The command IT_MacTCPVer has been made obsolete with the addition of the IT_TCPversion which provides better capability for determining version information across MacTCP, Open Transport and Winsock.

This command returns an Integer value into its first parameter indicating the version of MacTCP installed on the user's computer. The integer that is returned comes from the Apple Gestalt Manager and does not directly correspond to the specific MacTCP version string. This command is helpful in determining if any user is running the minimum required software needed to access many of the other commands.

versionCode - Currently, the Gestalt manager will return the following values corresponding to the MacTCP versions below. As Apple releases updates to its operating system and MacTCP, additional values may be returned.

Version Codes
0 = MacTCP driver not open
1 = v1.1
2 = v1.1.1
3 = v2.0

This command is most useful as a test of the users system to verify they have the necessary software (MacTCP) to use other commands within the 4D Internet Commands.

Example

The following function can be called at StartUp to set an Interprocess variable True or False, indicating if that user has the correct version of MacTCP installed.

```
    `Method: VERIFY_TCP ( {»Version} ) : Boolean [MacTCP installed]
C_INTEGER($ERR)
C_BOOLEAN($TCP_OK)
C_LONGINT(vTCPversion)
⇒ $ERR:=IT_MacTCPVer (vTCPversion)
$TCP_OK:=(($ERR=0)&(vTCPversion#0)) `TRUE = (No Error & known vers of MacTCP)
Case of
    : (Not($TCP_OK) & (<>SHOWERRORS)) `something's wrong & I should show errors
        ALERT("Problem opening your MacTCP driver. Verify MacTCP is installed
                                                                    properly.")
    : (($TCP_OK) & (Count parameters=1))  `everything's OK, return version
        LIST TO ARRAY("MacTCP Versions";$aTCPversion)
        If (vTCPversion<=Size of array($aTCPversion))
            $1»:=$aTCPversion{vTCPversion}
        Else
            $1»:="Unknown"
        End if  `(vTCPversion<=Size of array($aTCPversion))
End case
$0:=$TCP_OK  `return true if the version check was OK
```

IT_MyTCPAddr

IT_MyTCPAddr (ip_Address; subnet) → Integer

Parameter	Type		Description
ip_Address	String	←	IP address of users machine
subnet	String	←	Subnet Mask in IP form
Function result	Integer	←	Error Code

Description

The command IT_MyTCPAddr returns the IP address of the machine that executes the command.

ip_Address is the string returned which contains the IP address.

subnet is the string returned which contains the Subnet mask of the IP address.

IT_SetTimeOut

IT_SetTimeOut (timeout) → Integer

Parameter	Type		Description
timeout	Integer	→	Timeout in seconds; limited to 0 thru 127
Function result	Integer	←	Error Code

Description

The command IT_SetTimeOut sets the integer value of the timeout period in seconds. This value is limited to between zero and 127 seconds.

timeout is the current value in seconds of the timeout period. The following commands are affected by IT_SetTimeOut:

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
NET_Finger
NET_Ping
NET_Time

Note: Setting the timeout to zero for the TCP_Listen command allows it to listen indefinitely. Make sure you set the timeout back to some other value after this command, otherwise zero usually means use the default. Also, the timeout value is used for "TCP/IP timeouts" AND "wait for a response timeout". If you set the timeout to zero, it will never get enough time to wait for a response.

See Also

IT_GetTimeOut.

IT_GetTimeOut

version 6.5

IT_GetTimeOut (timeout) → Integer

Parameter	Type		Description
timeout	Integer	←	Timeout seconds
Function result	Integer	←	Error Code

Description

The command IT_GetTimeOut returns the current timeout value for the commands listed above in IT_SetTimeOut.

timeout is the current value in seconds of the timeout period.

See Also

IT_SetTimeOut.

IT_ErrorText

version 6.5

IT_ErrorText (error) → String

Parameter	Type		Description
error	Integer	→	Error code returned from other commands
Function result	String	←	Text of the error

Description

The command IT_ErrorText takes an integer error number as its only parameter and returns the String/Text description of that error. Note that this is one of the few 4D Internet Commands that does not return an Integer as its functional value.

error is the integer number of the error.

Example

The following is an example of an ErrorCheck routine that will display an alert message explaining the cause of an error.

```
	`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_Encode

IT_Encode (fileName; encodedFile; encodedMode) → Integer

Parameter	Type		Description
fileName	Text	→	A LocalPath to a file
encodedFile	Text	→	LocalPath file specification
		←	Path to resulting encoded file
encodedMode	Integer	→	1 = BinHex
			2 = Base64 (Data fork only)
			3 = AppleSingle
			4 = AppleDouble
			5 = AppleSingle AND Base64
			6 = AppleDouble AND Base64
			7 = UUEncode
			8 = MacBinary
Function result	Integer	←	Error Code

Description

The command IT_Encode encodes a file using the encodeMode specified. The specified file will not be altered and an encoded copy will be created. The name of the encoded file created will be the original file name plus a suffix appended to specify the encoding method. For Binhex encoding, the suffix ".bhx" will be appended. For Base64 encoding, the suffix ".b64" will be appended. For AppleSingle encoding, the suffix ".as" will be appended.

fileName takes a full pathname specification to a file you want to Encode. If an null string is passed in this parameter the user will be prompted with a dialog to select a file.

encodedFile can be passed a full LocalPath file specification providing a name and location for the encoded file. If not specified, the IT_Encode command will provide its own name for the document, placed in the same directory as the database structure. Whether specified or not, the final pathname of the encoded document will be returned in this parameter. Due to the potential for possible naming conflicts within the specified directory, you should always rely on the returned value as the true reference to the encoded file, not the original value passed into the command.

Note: If the original file name plus the suffix appended is greater than 31 characters, the command will truncate the original file name so that the new file name is not greater than the allowed 31 characters for a Macintosh file.

encodeMode identifies which encoding method to apply to the file. The default value is 1 for binhex encoding. Other methods are:

Code	**Scheme**
1	BinHex
2	Base64 (Data fork only)
3	AppleSingle
4	AppleDouble
5	AppleSingle and Base64
6	AppleDouble and Base64
7	UUEncode
8	MacBinary

When encoding using AppleDouble (encodeModes 4 & 6), two files are created named "%filename" and "filename".

See Also

IT_Decode.

IT_Decode

IT_Decode (fileName; decodedFile; decodeMode) → Integer

Parameter	Type		Description
fileName	Text	→	LocalPath to an encoded file
decodedFile	Text	→	LocalPath file specification
		←	Path of decoded file
decodeMode	Integer	→	1 = BinHex 2 = Base64 (Data fork only) 3 = AppleSingle 4 = AppleDouble 5 = AppleSingle AND Base64 6 = AppleDouble AND Base64 7 = UUEncode 8 = MacBinary
Function result	Integer	←	Error Code

Description

The command IT_Decode decodes a file using the decodeMode specified. The specified file will not be altered and a decoded copy will be created.

fileName takes a full pathname specification to a file you want to Decode. If an null string is passed in this parameter the user will be prompted with a dialog to select a file.

decodedFile can be passed a full LocalPath file specification providing a name and location for the decoded file. If not specified, the IT_Decode command will provide its own name for the document, placed in the same folder as the file specified in the first parameter. Whether specified or not, the full path of the decoded document will be returned in this parameter.

decodeMode identifies which decoding method to apply to the file. The default value is 1 for binhex decoding. Other methods are:

Code	Scheme
1	BinHex
2	Base64 (Data fork only)
3	AppleSingle
4	AppleDouble
5	AppleSingle and Base64
6	AppleDouble and Base64
7	UUEncode
8	MacBinary

When decoding using AppleDouble (decodeModes 4 & 6), this command looks for a file named "%filename" for the resource fork.

See Also

IT_Encode.

IT_GetPort

IT_GetPort (protocol; port) → Integer

Parameter	Type		Description
protocol	Integer	→	1 = FTP; 2 = SMTP; 3 = POP3
port	Integer	←	Port Number
Function result	Integer	←	Error Code

Description

Given a specified protocol, the command IT_GetPort will get the current port number being used by the 4D Internet Commands related to the protocol.

See Also

IT_SetPort.

IT_SetPort

IT_SetPort (protocol; port) → Integer

Parameter	Type		Description
protocol	Integer	→	1 = FTP; 2 = SMTP; 3 = POP3
port	Integer	→	Port Number
Function result	Integer	←	Error Code

Description

Given a specified protocol, the command IT_SetPort will direct all future communication of the protocol to the specified port.

See Also

IT_GetPort.

9 Appendixes

Executing Commands via a Case Statement

In many of the examples in this document and throughout the example database, a programming construct is used which is likely to be unfamiliar to many developers. Many of these examples execute a series of commands as falsified cases within a 4th Dimension Case statement.

Many of the commands within 4D Internet Commands require that an entire sequence of commands execute successfully in order to complete. Since the failure of any one command within the sequence should stop any further processing along that path, it would become laborious to cascade your If conditions many level deeps:

```
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)
            …and deeper, and deeper…
         End if
      End if
   End if
End if
```

An alternative to this method is to rely on the manner in which 4D executes it's case statements. Each item of a case statement is executed by 4D in order to determine if its return value is **True** or **False**. If all elements of a case statement were to return a false value, then each element of that case statement's tests would have been run. We can execute the same code described above by the following:

```
$SentOK:=False  `A flag to indicate if we made it through all of the calls
Case of
   : (SMTP_New ($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
   $SentOK:=True  `message was composed and mailed successfully
End case
If ($smtp_id#0)  `If a Message Envelope was created we should clear it now
   $OK:=SMTP_Clear ($smtp_id)
End if
```

In the above example, every 4D Internet command will return a zero error number if it successfully completed. In order for 4D to evaluate each case statement, it must actually execute each external call to obtain its return value. Since each case element compares the return result to not zero, 4th Dimension will not find an element to stop on until one of the commands fails. If every command executes successfully, 4D will proceed down to run the **Else** condition where we set the *$SentOK* flag to indicate that the message was composed and sent successfully.

Suggestions when auto-replying to POP3 mail

If you are planning on implementing a mail system within your database in which the user can "Reply" to mail they have received, there are some standard suggestions for how to fill out the fields of the reply message. The following suggestions are outlined by RFC#822:

• The address listed in the "Sender" field should receive notices of any problems during delivery of the initial messages. If no "Sender" field exists, notices should be sent to the address listed in the "From" field. The "Sender" mail address should only be sent replies pertaining to problems in the mail delivery and **not** replies related to the topic of the message.

• The "Sender" address should never be used in an automated process of replying to messages. Instead, the message should either use the "Reply-To" field or the "From" field, dependent on the conditions described below.

• If the "Reply-To" field exists and contains one or more mail addresses, then any reply should be directed to the people in that list. Addresses within the "Reply-To" header override any addresses listed in the "From" header. However, if no "Reply-To" field exists but a "From" field does exist, replies should be sent to the mailbox(es) indicated in the "From" header.

These suggestions are only meant to help the decision process when the mail addressing is programmatically handled in the case of "Reply" type actions. Once the Reply message has been created, the end-user can certainly override any of these defaults before sending the message.

How to choose a port number

• 0 to 1023 (Well Known Ports): The Well Known Ports are assigned by the I.A.N.A. (Internet Assigned Numbers Authority) and on most systems can only be used by system (or root) processes or by programs executed by privileged users.

- 20 and 21 FTP;
- 23 TELNET;
- 25 SMTP;
- 37 NTP;
- 80 and 8080 HTTP.

• 1024 to 49151 (Registered Ports): The Registered Ports are listed by the I.A.N.A. and on most systems can be used by ordinary user processes or programs executed by ordinary users (routers, specific applications...)

• 49152 to 65535 (Dynamic and/or Private Ports) : The Dynamic and/or Private Ports are free of use.

People who want to use TCP/IP commands to synchronize databases would have to use port numbers higher than 49151.

For more information, please visit the I.A.N.A. Web site: http://www.iana.org

TCP Port Numbers

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
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

All 4D Internet Commands (with the exception of IT_ErrorText & IT_Version) return an integer value as the result of the function. This integer contains any error number which the command needs to convey back to the 4D database. If a command is successful, a zero will be returned. The source of an error number can usually be determined by the range of values which the error falls within. The following table provides an index to the most likely creator of an error in any given range:

Error Number	**Generated by**
Error < Zero	Operating System Error
Zero	No Error
Error >= 10000	4D Internet Commands Error

4D Internet Commands Error Codes

Iif an error occurs during any operation, a numeric value from the following table will be returned:

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 a 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 a 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

Open Transport Error Codes

-3211	Open Transport Out of Memory
-3201	Open Transport Not Found
-3216	Open Transport duplicate found
-3150	A Bad address was specified
-3151	A Bad option was specified
-3152	Missing access permission
-3153	Bad provider reference
-3154	No address was specified
-3155	Call issued in wrong state
-3156	Sequence specified does not exist
-3157	A system error occurred
-3158	An event occurred - call Look()
-3159	An illegal amount of data was specified
-3160	Passed buffer not big enough
-3161	Provider is flow-controlled
-3162	No data available for reading
-3163	No disconnect indication available
-3164	No Unit Data Error indication available
-3165	A Bad flag value was supplied
-3166	No orderly release indication available
-3167	Command is not supported
-3168	State is changing - try again later
-3169	Bad structure type requested for OTAlloc
-3170	A bad endpoint name was supplied
-3171	A Bind to an in-use addr with qlen > 0
-3172	Address requested is already in use
-3173	Accept failed because of pending listen
-3174	Tried to accept on incompatible endpoint
-3175	kOTResQLenErr
-3176	kOTResAddressErr
-3177	kOTQFullErr

-3178	An unspecified provider error occurred
-3179	A synchronous call at interrupt time
-3180	The command was cancelled
-3200	Permission denied
-3201	No such file or directory
-3202	No such resource
-3203	Interrupted system service
-3204	I/O error
-3205	No such device or address
-3208	Bad file number
-3210	Try operation again later
-3211	Not enough space
-3212	Permission denied
-3213	Bad address
-3215	Device or resource busy
-3216	File exists
-3218	No such device
-3221	Invalid argument
-3224	Not a character device
-3231	Broken pipe
-3233	Message size too large for STREAM
-3234	Call would block, so was aborted
-3234	or a deadlock would occur
-3236	kEALREADYErr
-3237	Socket operation on non-socket
-3238	Destination address required
-3239	Message too long
-3240	Protocol wrong type for socket
-3241	Protocol not available
-3242	Protocol not supported
-3243	Socket type not supported
-3244	Operation not supported on socket
-3247	Address already in use
-3248	Can't assign requested address
-3249	Network is down
-3250	Network is unreachable
-3251	Network dropped connection on reset
-3252	Software caused connection abort
-3253	Connection reset by peer
-3254	No buffer space available
-3255	Socket is already connected
-3256	Socket is not connected
-3257	Can't send after socket shutdown
-3258	Too many references: can't splice
-3259	Connection timed out
-3260	Connection refused
-3263	Host is down
-3264	No route to host

-3269	kEPROTOErr
-3270	kETIMEErr
-3271	kENOSRErr
-3272	kEBADMSGErr
-3273	kECANCELErr
-3274	kENOSTRErr
-3275	kENODATAErr
-3276	kEINPROGRESSErr
-3277	kESRCHErr
-3278	kENOMSGErr
-3279	kOTClientNotInittedErr
-3280	kOTPortHasDiedErr
-3281	kOTPortWasEjectedErr
-3282	kOTBadConfigurationErr
-3283	kOTConfigurationChangedErr
-3284	kOTUserRequestedErr
-3285	kOTPortLostConnection

MacTCP & Miscellaneous Error Codes

-33	Unable to write to the disk. The File Directory is full.
-34	Unable to write to the disk. The disk is full.
-35	No such volume.
-36	I/O error.
-37	Bad file name or volume name.
-38	File is not open for reading or writing.
-39	Unable to read from the file. The end of file was reached.
-42	Unable to continue because too many files are open.
-43	The file cannot be found.
-44	The volume is locked at the hardware level.
-45	The file is locked.
-46	The volume is locked at the software level.
-47	One or more files are already open by another application.
-48	A file with this name already exists.
-49	File already open with write permission.
-54	Permissions error on opened file.
-57	Not a Macintosh volume.
-59	An error occured while trying to rename the file.
-61	Write permissions error.
-108	Insufficient amount of memory.
-120	Directory not found.
-23000	Unable to initialize the local network handler.
-23001	The manually set address is configured improperly.
-23002	A configuration resource is missing.
-23003	Not enough room in the application heap to load MacTCP.
-23004	Error in getting an address from a server or the address is already in use by another machine.

-23005	A TCPClose command was already issued so there is no more data to send on this connection.
-23006	The total amount of data described by the WDS was either 0 or greater than 65,535 bytes.
-23007	The TCP or UDP stream already has an open connection.
-23008	This TCP stream has no open connection.
-23009	Maximum TCP or UDP streams are already open.
-23010	The specified TCP or UDP stream is not open.
-23011	An open stream is already using this receive buffer area.
-23012	The TCP connection was broken.
-23013	The receive buffer area pointer is 0.
-23014	Invalid RDS or WDS buffers.
-23015	The connection came halfway up and then failed.
-23016	The specified command action was not completed in the specified time period.
-23017	A stream is already open using this local UDP port or a TCP connection already exists between this local IP address and TCP port, and the specified remote IP address and TCP port.
-23032	The packet is too large to send without fragmenting and the Don't Fragment flag is set.
-23033	The destination host is not responding to address resolution requests.
-23035	The icmp echo packet was not responded to in the indicated timeout period.
-23036	Insufficient internal driver buffers available to fragment this packet on send.
-23037	No gateway available to manage routing of packets to off-network destinations.
-23041	The hostName field had a syntax error. The address was given in dot notation (that is, W.X.Y.Z) and did not conform to the syntax for an IP address.
-23042	The name specified cannot be found in the cache. The domain name resolver will now query the domain name server and return the answer in the call-back procedure.
-23043	No result procedure is passed to the address translation call when the resolver must be used to find the address.
-23044	No name server can be found for the specified name string.
-23045	This domain name does not exist.
-23046	None of the known name servers are responding.
-23047	The domain name server has returned an error.
-23048	Not enough memory is available to issue the needed DNR query or to build the DNR cache.

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 Sockects 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 Sockects 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 were 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 (for example, 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.

The references below contain WWW (World Wide Web) pointers to additional sources of information related to the internet protocols. Web documents may be accessed via programs like Netscape, Internet Explorer or Mosaic.

http://rs.internic.net/
To understand what is a Domain Name and what you have to do to register one.

http://www.ietf.org/
Internet Engineering Task Force (IETF) site.

http://www.rfc-editor.org/
To understand what is an RFC and to search for RFCs and sites related to the RFC series (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/
All you need to know about the World Wide Web.

http://www.con.wesleyan.edu/~triemer/network/docservs.html
Listing of all TCP/IP port numbers with pointers to additional sources of information on many of the protocols. If you're going to work with the low-level TCP routines this is a site to become familiar with.

Command Index