



Preface

version 2004 (Modified)

The 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 more and more 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 and IMAP 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 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 Commands allow 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 that will connect either to POP3 (Post Office Protocol, Version 3) or IMAP (Internet Message Access Protocol) mail servers for retrieval of mail messages and encoded attachments. Since the suite of SMTP, POP3 and IMAP commands conforms 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 provides developers with the essential tools to build and control their own Internet communications. In addition, the TCP_Open command allows connection using a SSL (Secured Socket Layer) protocol.

Some examples are:

- · 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
- Connect to a Web server in Https.

Note: For greater flexibility, 4D Internet commands let you pass a POP3, IMAP or FTP connection reference directly to low-level TCP commands and vice versa. For more information, refer to the Low Level Routings. Overview section

Installation and Software Requirements

version 2004 (Modified)

Installation

The "4D Internet Commands" plug-in is integrated in 4th Dimension in the same way as other plug-ins.

4D Internet Commands becomes automatically available when you install a 4D product since the plug-in is automatically installed in the PlugIns folder of your application.

For more information on plug-in installation and configuration, please refer to the 4D Product Line 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 that 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.

System

The system requirements are the same as those for 4th Dimension. For more information, please refer to the 4D Product Line Installation Guide.

4th Dimension (Mac & Windows)

- 4D for Mac version 2004 or higher
- 4D for Windows version 2004 or higher

BSD, 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, examples of which are BSD Sockets on the Macintosh and Winsock on Windows.

For more information on configuring TCP/IP, please refer to your network administrator.

Network access

In order to use the suite of commands in 4D Internet Commands, you must have access to a network that 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, please refer to your network administrator.

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.

IMAP Mail Server

In order to use the IMAP commands, you must have an account on an IMAP mail server.

Glossary and Terminology

version 6

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 that could only be effectively carried out if controlled by a single agency. The NIC is the group responsible for such administrative reads.

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. <u>Appendix D</u>, <u>Additional Information...</u> 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 suith 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.4D.com" or "ftp.4D.com".

Domain Name = "4D.com"

Host Name (Name of a computer)

= IP address = ip_LongInt

"www.4D.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@4D.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 "4D.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 names and 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 4D.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 that 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 DeluxeTM Compact ProTM or WinZipTM. 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.STT 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 transferred from machine to machine on its way to its ultimate destination.

The descriptions that follow provide details on the meaning and formatting of many key parameters used throughout this manual.

Parameter	Туре		Description	
hostName	String	\rightarrow	Host name (Ex: "www.companyname.com")	
			IP address (Ex: "204.118.90.2")	
ip_LongInt	LongInt	\rightarrow	Long Integer reference to a IP address	
mailAddress	Text	\rightarrow	Ex: "jsmith@4d.com"	
addressList	Text	\rightarrow	Ex: "jsmith@4d.com, jdupont@4d.fr" or	
			"jsmith@4d.com"+Char(13)+"jdupont@4d.fr"	
localPath	Text	\rightarrow	- 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	\rightarrow	- Document	
			"/usr/jsmith/reports/salesreport.txt"	
			- Directory	
			"/usr/jsmith/reports/"(Note trailing "/")	
tcp_ID	LongInt	\rightarrow	Reference to an open TCP session	
smtp_ID	LongInt	\rightarrow	Reference to a new mail message	
pop3_ID	LongInt	\rightarrow	Reference to an open POP3 session	
imap_ID	Longint	\rightarrow	Reference to an open IMAP connection	
ftp_ID	LongInt	\rightarrow	Reference to an open FTP session	

Function result Integer \leftarrow Error Code

hostName

The hostName is the host name or IP address, such as "dns.4d.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.4d.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 LongIn

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. However, for compatibility reasons with IPV6, 4D advises developers against using this feature.

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 address. It 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:

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

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:

localPath

The localPath is the location of a file or directory on the users machine (Mac or Windows). On a Macintosh, colons delimit items within folders. 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.

hostPatl

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 "4D" directory would be specified as "/4D/reports/report.txt". A directory pathname should end with a "/" character. Note that a full pathname begins with a "/" which reports of the volume.

smtp_ID, pop3_ID, imap_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, IMAP, 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_Clear
pop3_ID	POP3_Login	POP3_Logout or POP3_VerifyID
imap_ID	IMAP_Login	IMAP_Logout or IMAP_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 that 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.

IC Send Mail

			11/1/20	
Sending Mail, Overview	_			
 SMTP_SetPrefs (lineFeed; book 	yType; lineLength) → Integer dyType; lineLength) → Integer			
SMTP_GetPres (lineFeeds; b SMTP_QuickSend (hostName)	msgFrom; msgTo; subject; message) - ger	→Integer		
 SMTP_New (smtp_ID) → Int SMTP_Host (smtp_ID: hostNa 	eger me{; deleteOption}) → Integer			
$SMTP_Send$ (smtp_ID) \rightarrow In	eger			
SMTP_Clear (smtp_ID) SMTP_Date (smtp_ID; msgDa	e: msgTime: timeZone: offset{: delete	Option}) → Integer		
SMTP_From (smtp_ID; msgF SMTP_Sender (smtp_ID; msg	om{; deleteOption}) → Integer Sender{; deleteOption}) → Integer			
<u>SMTP_ReplyTo</u> (smtp_ID; rep.	lyTo{; deleteOption}) → Integer			
	py{; deleteOption}) → Integer			
SMTP_Bcc (smtp_ID; blindCa SMTP_InReplyTo (smtp_ID;	bon{; deleteOption}) → Integer nReplyTo{; deleteOption}) → Integer			
SMTP_References (smtp_ID;	nReplyTo{; deleteOption}) Integer eferences{; deleteOption}) Integer omments{; deleteOption}) Integer omments{; deleteOption})			
SMTP_Keywords (smtp_ID; l	eywords{; deleteOption}) -> Integer			
SMTP_Encrypted (smtp_ID; of SMTP_AddHeader (smtp_ID; of SMTP_Note: Smtp_ID; of SMTP_ID; o	headerName; headerText{; deleteOptic	on}) → Integer		
SMTP_Subject (smtp_ID; sub SMTP_Body (smtp_ID; msgB	ect{; deleteOption}) → Integer dy{; deleteOption}) → Integer			
SMTP_Attachment (smtp_ID; SMTP_Charset (encodeHeade	fileName; encodeType{; deleteOption}) → Integer		
	me; password{; authMode}) \longrightarrow Integer	er		
Other related commands:				
Other related commands:				
MSG_FindHeader (fileName;	neaderLabel; headerValue) Integ	ger Theme: <u>IC Downloaded Mail</u>		
POP3_Charset (decodeHeader	s; bodyCharset) Integer Them	e: IC POP3 Review Mail		
IT_Encode (fileName; encode	File; encodedMode)Integer T	Theme: <u>IC Utilities</u>		

Sending Mail, Overview

version 6.5

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 email 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

At least one address in this header

• <u>SMTP_To</u>

At least one address in this header

SMTP_Send

Sends the message

• SMTP_Clear

Clears any memory used during the creation of the message $% \left(1\right) =\left(1\right) \left(1\right) \left($

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 (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,
lineLength	Longint -	÷	 -1 = No Change) Maximum line length (0 = [default] Autodetect, -1 = No Change)

Function result Integer ← Error Code

Description

The SMTP_SetPrefs command sets the preferences of messages to be sent using the SMTP commands. The command has a global scope and will affect all subsequent messages created with the SMTP commands. The configurable options affect the format of a mail message as it is sent to a SMTP server using the SMTP_QuickSend or SMTP_Send commands. 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/naragraph 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 character set used in the message body to be sent (Body-Content-Type) as well as the encoding to apply to the message body (Content-Transfer-Encoding), according to the values in the table below. For example, "US-ASCII & 7 bit" (value 2) means that the message body charset currently in use is supposed to be US ASCII — includes only standard ASCII codes (0 through 127) which are common to Windows and Macintosh—and that 4D IC will encode the message body using the 7 bit encoding. Note that the SMTP_SetPres command does NOT convert the message body using the specified character set, this has to be managed by the user if necessary. If you want to force the character set conversion, refer to the SMTP_Charset command description.

If not changed, the default content type is 1, which will allow 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
- 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
- 9 Reserved
- 10 ISO-2022-JP (Japanese) & 7 bit
- 11 ISO-2022-KR (Korean) & 7 bit

- 12 ISO-2022-CN (Traditional & Simplified Chinese) & 7 bit
- 13 HZ-GB-2312 (Simplified Chinese) & 7 bit
- 14 Shift-JIS (Japanese) & base64

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_Charset, SMTP_GetPrefs.

SMTP_GetPrefs

version 6.5

 $SMTP_GetPrefs \ (lineFeeds; bodyType; lineLength) \longrightarrow Integer$

Parameter	Type		Description
lineFeeds	Integer	\leftarrow	0 = Don't Add, 1 = Add LineFeeds
bodyType	Integer	\leftarrow	Body-Content-Type
lineLength	Longint	\leftarrow	Maximum line length

Function result Integer \leftarrow Error Code

Description

The SMTP_GetPrefs command 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.

 ${\it line Feeds} \ {\it returns} \ {\it the current setting determining how the SMTP commands will handle carriage returns within the body of a message.}$

 $body Type \ \text{returns the current setting for the Body-Content-Type. See the } body Type \ \text{table under } \underline{SMTP_SetPrefs} \ \text{for a description of the values}.$

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

See Also

SMTP_SetPrefs.

 $SMTP_QuickSend~(hostName;~msgFrom;~msgTo;~subject;~message) \longrightarrow Integer$

Parameter	Type		Description
hostName	String	\rightarrow	Host name or IP address
msgFrom	Text	\rightarrow	MailAddress or AddressList
msgTo	Text	\rightarrow	MailAddress or AddressList
subject	Text	\rightarrow	Subject
message	Text	\rightarrow	Message

Function result Integer \leftarrow Error Code

Description

The SMTP_QuickSend command 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: Usually, the subject of the message should not contain characters with diacritical marks (such as é, ö, etc.). However, if you want to use such "extended" characters, refer to the SMTP_Charset command descriptions.

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.4d.com"
$ToAddress:="adupont@4d.fr"
$FromAddress:="jsmith@4d.com"
$Subject:="Sales Report"
$Message:="Can you send me the sales report for January 2000? Thanks."
$Error:=$MTP_QuickSend ($Host;$FromAddress;$ToAddress;$Subject;$Message)
If ($Error#0)
    ALERT("Error: SMTP_QuickSend"+ Char(13)+IT ErrorText ($Error))
End If
```

See Also

SMTP_Charset, SMTP_New, SMTP_SetPrefs.

SMTP	Mari

version 6.5

SMTP_New (smtp_ID) \longrightarrow Integer

ParameterTypeDescriptionsmtp_IDLongint←Reference to this new message

Function result Integer \leftarrow Error Code

Description

The SMTP_New command 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.

Examples

See the examples for the command <u>SMTP_Body</u> and <u>SMTP_Send</u>.

See Also

SMTP_Clear, SMTP_QuickSend, SMTP_Send.

SMTP_Host

version 6.5

 $SMTP_Host (smtp_ID; hostName\{; deleteOption\}) \longrightarrow Integer$

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
hostName	String	\rightarrow	Host name or IP address
deleteOption	Integer	\rightarrow	0 = Add or Replace, 1 = Delete

Function result Integer \leftarrow 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 $\underline{SMTP_New}$ 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.

Examples

See the examples for the command <u>SMTP_Body</u> and <u>SMTP_Send</u>.

See Also

version 6.5

SMTP_Send (smtp_ID) \rightarrow Integer

ParameterTypeDescriptionsmtp_IDLongint→ Message reference

Function result Integer \leftarrow Error Code

Description

The SMTP_Send command 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)
$error:=\frac{SMTP_Host}{Serror:=\frac{SMTP_From}{Serror:=\frac{SMTP_From}{Serror:=\frac{SMTP_ReplyTo}{Semtp_id;"herb_tarlick@wkrp.com"}}$
$error:=SMTP_Subject ($smtp_id;"Discounts on Ad Space!")
FIRST RECORD([People])
For($i;1; Records in Selection([People]))
   If ([People]Sales2Date>100000)
      $Body:=<>BigDiscText
   Else
      $Body:=<>SmlDiscText
   End if
   $Body:=Replace String ($BoilerPlate;"<Salutation>";[People]Firstname)
   $error:=SMTP_Body ($smtp_id;$Body)
   $error:=SMTP_Send ($smtp_id)
   NEXT RECORD([People])
End for
$error:=SMTP_Clear ($smtp_id)
```

See Also

SMTP_Clear (smtp_ID) \longrightarrow Integer

Parameter Type smtp_ID Longint

Description

- Message reference
- ← 0 if successful

Function result Integer \leftarrow Error Code

Description

The SMTP_Clear command 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 <u>SMTP_New</u> command. Upon the successful close of a SMTP message, the <u>SMTP_Clear</u> command will return a zero value back into the smtp_ID variable.

Example

See the example for the command $\underline{SMTP_Body}$.

See Also

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

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

Function result Integer \leftarrow Error Code

Description

Given a date, a time, and a geographical location of the mail creator, the SMTP_Date command 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 $\underline{SMTP_New}$ 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.

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 time zone of a Macintosh computer is determined by the settings in the **Date&Time** system preferences. 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 Military Time table below determines offset. 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 From

version 6.5

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

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
msgFrom	Text	\rightarrow	MailAddress or AddressList
deleteOption	Integer	\rightarrow	0 = Add, $1 = Replace$, $2 = Delete$

Function result Integer \leftarrow Error Code

Description

The SMTP_From command 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 mandatory. 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 <u>SMTP_New</u> 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 whether to add or delete the "From" header:

- A value of zero will add the new value to the "From" field.
- A value of 1 will set the "From" field to the new value, overriding any prior settings (if you pass an empty string in msgFrom, the header will be removed from the mail envelope).
- · A value of 2 will delete any address previously defined for the "From" 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 people compose a message on the subject of a company policy change that 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:=\final SMTP_From (\$\text{smtp_id};\$From;\text{0})
$Error:=\final SMTP \text{Subject} (\$\text{smtp_id};\"Company Policy Change";\text{0})
$Error:=\final SMTP \text{To} (\$\text{smtp_id};\$<>All\text{Employee};\text{0})
```

See Also

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

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
msgSender	Text	\rightarrow	MailAddress (1 only)
deleteOption	Integer	\rightarrow	0 = Add, $1 = Replace$, $2 = Delete$

Function result Integer \leftarrow Error Code

Description

The <u>SMTP_Sender</u> command 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 <u>SMTP_New</u> command.

 $msgSender\ contains\ a\ single\ Mail Address\ to\ be\ listed\ in\ the\ Sender\ field\ of\ the\ message.\ Only\ \textbf{one}\ mail\ address\ may\ be\ specified\ for\ this\ header.$

deleteOption is an integer value which specifies whether to add or delete the Sender header:

- A value of zero will add the new value to the Sender field.
- A value of 1 will set the Sender field to the new value, overriding any prior settings (if you pass an empty string in msgSender, the header will be removed from the mail envelope).
- A value of 2 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:=\frac{SMTP_From}{\$\smtp_id;\$\From;0}
$Error:=\frac{SMTP_Sender}{\$\smtp_id;\$\secretary@acme.com\$\;0)
$Error:=\frac{SMTP_Subject}{\$\smtp_id;\$\Company Policy Change\$\;0)
$Error:=\frac{SMTP_To}{\$\smtp_id;\$\Smtp_id;\$\Smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\
```

See Also

 $SMTP_ReplyTo (smtp_ID; replyTo\{; deleteOption\}) \longrightarrow Integer$

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

Function result Integer \leftarrow Error Code

Description

The SMTP_ReplyTo command 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 affect 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 $\underline{SMTP_New}$ 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 that 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:=\frac{SMTP_From}{\$\smtp_id;\$\smtp_id;\$\secretary@acme.com";0}
$Error:=\frac{SMTP_Sender}{\$\smtp_id;\$\secretary@acme.com, personnel_dept@acme.com";0)}
$Error:=\frac{SMTP_Subject}{\$\smtp_id;\$\company Policy Change";0)}
$Error:=\frac{SMTP_To}{\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_id;\$\smtp_i
```

See Also

 $SMTP_To (smtp_ID; msgTo\{; deleteOption\}) \longrightarrow Integer$

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
msgTo	Text	\rightarrow	MailAddress or AddressList
deleteOption	Integer	\rightarrow	0 = Add, $1 = Replace$, $2 = Delete$

Function result Integer \leftarrow Error Code

Description

The SMTP_To command adds the identity of the primary recipients of the message. 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 whether to add or delete the "To" header:

- A value of zero will add the new value to the "To" field.
- A value of 1 will set the "To" field to the new value, overriding any prior settings (if you pass an empty string in msgTo, the header will be removed from the mail envelope).
- A value of 2 will delete any address previously defined for the "To" field and remove the header from the mail envelope.

deleteOption is an optional parameter which will default to zero if not otherwise specified.

Example

See the example for the command <u>SMTP_Body</u>.

See Also

 $SMTP_Cc \ (smtp_ID; carbonCopy\{; deleteOption\}) \longrightarrow Integer$

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
carbonCopy	Text	\rightarrow	MailAddress or AddressList
deleteOption	Integer	\rightarrow	0 = Add, $1 = Replace$, $2 = Delete$

Function result Integer \leftarrow Error Code

Description

The <u>SMTP_Cc</u> command adds carbon copy recipients to the message specified by <u>smtp_ID</u>. It is not mandatory to have any addresses in the Cc: field. 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 $\underline{SMTP_New}$ command.

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

deleteOption is an integer value which specifies whether to add or delete the "Cc" header:

- \bullet A value of zero will add the new value to the "Cc" field.
- A value of 1 will set the "Cc" field to the new value, overriding any prior settings (if you pass an empty string in carbonCopy, the header will be removed from the mail envelope).
- A value of 2 will delete any address previously defined for the "Cc" field and remove the header from the mail envelope.

deleteOption is an optional parameter which will default to zero if not otherwise specified.

Example

See the example for the command **SMTP_Body**.

See Als

SMTP_Bcc, SMTP_New.

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

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
blindCarbon	Text	\rightarrow	AddressList
deleteOption	Integer	\rightarrow	0 = Add, $1 = Replace$, $2 = Delete$

Function result Integer ← Error Code

Description

The SMTP_Bcc command adds blind carbon copy recipients to the message specified by smtp_ID. It is not mandatory to have any addresses in the Bcc: field.

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 whether to add or delete the "Bcc" header:

- A value of zero will add the new value to the "Bcc" field.
- A value of 1 will set the "Bcc" field to the new value, overriding any prior settings (if you pass an empty string in blindCarbon, the header will be removed from the mail envelope).
- A value of 2 will delete any address previously defined for the "Bcc" 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 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 address is added to the blind carbon copy list.

See Also

SMTP_Cc, SMTP_New.

SMTP_InReplyTo

version 6.5

 $SMTP_InReplyTo\left(smtp_ID; inReplyTo\{; deleteOption\}\right) \longrightarrow Integer$

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
inReplyTo	Text	\rightarrow	In-Reply-To Text
deleteOption	Integer	\rightarrow	0 = Replace (if inReplyTo not empty),
			1 = Replace, 2 = Delete

Function result Integer ← Error Code

Description

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

 $smtp_ID$ is the long integer reference to the mail message created with the $\underline{SMTP_New}$ 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 integer value which specifies whether to replace or delete the "ReplyTo" header:

- A value of zero will set the "ReplyTo" field to the new value, overriding any prior settings (if you pass an empty string in inReplyTo, the prior header will be kept).
- A value of 1 will set the "ReplyTo" field to the new value, overriding any prior settings (if you pass an empty string in inReplyTo, the header will be deleted).
- A value of 2 will delete any address previously defined for the "ReplyTo" field and remove the header from the mail envelope.

deleteOption is an optional parameter which will default to zero if not otherwise specified.

See Also

SMTP_References

version 6.5

 $SMTP_References (smtp_ID; references\{; deleteOption\}) \longrightarrow Integer$

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
references	Text	\rightarrow	Reference Text
deleteOption	Integer	\rightarrow	0 = Replace (if references not empty),
			1 = Replace, 2 = Delete

Function result Integer \leftarrow Error Code

Description

The SMTP_References command identifies additional correspondences that the message references.

 $smtp_ID$ is the long integer reference to the mail message created with the $\underline{SMTP_New}$ 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 integer value which specifies whether to replace or delete the "References" header:

- A value of zero will set the "References" field to the new value, overriding any prior settings (if you pass an empty string in references, the prior header will be kept).
- A value of 1 will set the "References" field to the new value, overriding any prior settings (if you pass an empty string in references, the header will be deleted).
- A value of 2 will delete any reference previously defined for the "References" field and remove the header from the mail envelope.

deleteOption is an optional parameter which will default to zero if not otherwise specified.

See Also

version 6.5

 $SMTP_Comments \ (smtp_ID; comments\{; deleteOption\}) \longrightarrow Integer$

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
comments	Text	\rightarrow	Comment text
deleteOption	Integer	\rightarrow	0 = Replace (if comments not empty),
			1 = Replace, 2 = Delete

Function result Integer ← Error Code

Description

The SMTP_Comments command 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 <u>SMTP_New</u> command.

 ${\it comments}$ 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.

 ${\it deleteOption} \ {\rm is\ an\ integer\ value\ which\ specifies\ whether\ to\ replace\ or\ delete\ the\ "Comments"\ header:}$

- A value of zero will set the "Comments" field to the new value, overriding any prior settings (if you pass an empty string in comments, the prior header will be kept).
- A value of 1 will set the "Comments" field to the new value, overriding any prior settings (if you pass an empty string in comments, the header will be deleted).
- $\bullet \ A \ value \ of \ 2 \ will \ delete \ any \ comments \ previously \ defined \ for \ the \ "Comments" \ field \ and \ remove \ the \ header \ from \ the \ mail \ envelope.$

 ${\it deleteOption} \ {\rm is\ an\ optional\ parameter\ which\ will\ default\ to\ zero\ if\ not\ otherwise\ specified.}$

Example

See the example for the command $\underline{\mathit{SMTP_Body}}$.

See Also

SMTP_Keywords

version 6.5

 $SMTP_Keywords (smtp_ID; keywords\{; deleteOption\}) \longrightarrow Integer$

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
keywords	Text	\rightarrow	Keywords List
deleteOption	Integer	\rightarrow	0 = Replace (if keywords not empty),
			1 = Replace, 2 = Delete

Function result Integer ← Error Code

Description

The SMTP_Keywords command is used to insert keywords into the "Keywords" header of the message designated by the smtp_ID field.

 $smtp_ID$ is the long integer reference to the mail message created with the $\underline{SMTP_New}$ 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 integer value which specifies whether to replace or delete the "Keywords" header:

- A value of zero will set the "Keywords" field to the new value, overriding any prior settings (if you pass an empty string in keywords, the prior header will be kept).
- A value of 1 will set the "Keywords" field to the new value, overriding any prior settings (if you pass an empty string in keywords, the header will be deleted).
- A value of 2 will delete any keywords previously defined for the "Keywords" field and remove the header from the mail envelope.

deleteOption is an optional parameter which will default to zero if not otherwise specified.

See Also

version 6.5

SMTP_Encrypted (smtp_ID; encrypted{; deleteOption})

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
encrypted	Text	\rightarrow	Encryption method
deleteOption	Integer	\rightarrow	0 = Replace (if encrypted not empty),
			1 = Replace, 2 = Delete

Description

The <u>SMTP_Encrypted</u> command 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 <u>SMTP_Body</u>), 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 $\underline{SMTP_New}$ command.

encrypted is a text value which specifies the type of encryption method used to encrypt the body of the message. The recipients mail software, which determines the method needed to decrypt the message body uses the encrypted header. 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.

 ${\it deleteOption} \ is \ an integer \ value \ which \ specifies \ whether \ to \ replace \ or \ delete \ the \ "Encrypted" \ header:$

- A value of zero will set the "Encrypted" field to the new value, overriding any prior settings (if you pass an empty string in encrypted, the prior header will be used).
- A value of 1 will set the "Encrypted" field to the new value, overriding any prior settings (if you pass an empty string in encrypted, the header will be deleted).
- A value of 2 will remove the "Encrypted" field from the mail envelope.

 ${\it deleteOption} \ is \ an \ optional \ parameter \ which \ will \ default \ to \ zero \ if \ not \ otherwise \ specified.$

See Also

SMTP_Body, SMTP_New.

SMTP_AddHeader (smtp_ID; headerName; headerText{; deleteOption}) -> Integer

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
headerName	String	\rightarrow	Name of header
headerText	Text	\rightarrow	Header text
deleteOption	Integer	\rightarrow	0 = Add
			1 = Replace all headers with 'headerName',
			2 = Remove all headers named 'headerName'

Function result Integer \leftarrow Error Code

Description

The <u>SMTP_AddHeader</u> command allows users to add their own headers to the message referenced by <u>smtp_ID</u>. 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 <u>SMTP_AddHeader</u> 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 MSG_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 that 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 $\underline{SMTP_New}$ 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

 $To send \ a \ HTML \ encoded \ message, you should \ insert \ HTML \ tags \ in \ the \ message \ body \ (i.e. < HTML>, < HEAD>, \ etc.) \ and \ replace \ all \ "Content-Type" \ headers \ with \ the \ string \ "text/html; charset=us-ascii":$

See Also

MSG_FindHeader, SMTP_New.

version 6.5

 $SMTP_Subject (smtp_ID; subject\{; deleteOption\}) \longrightarrow Integer$

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
subject	Text	\rightarrow	Subject of message
deleteOption	Integer	\rightarrow	0 = Replace (if subject not empty
			1 = Replace, 2 = Delete

Function result Integer \leftarrow Error Code

Description

The SMTP_Subject command adds the subject of 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 <u>SMTP_New</u> command.

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

Warning: Usually, the subject of the message should not contain characters with diacritical marks (such as é, ö, etc.). However, if you want to use such "extended" characters, refer to the SMTP_SetPrefs and SMTP_Charset command descriptions.

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 integer value which specifies whether to replace or delete the "Subject" header:

- A value of zero will set the "Subject" field to the new value, overriding any prior settings (if you pass an empty string in subject, the prior header will be used).
- A value of 1 will set the "Subject" field to the new value, overriding any prior settings (if you pass an empty string in subject, the header will be deleted).
- A value of 2 will remove the "Subject" field from the mail envelope.

 ${\it deleteOption} \ is \ an \ optional \ parameter \ which \ will \ default \ to \ zero \ if \ not \ otherwise \ specified.$

Example

See the example for the command <u>SMTP_Body</u>.

See Also

SMTP_Charset, SMTP_New, SMTP_SetPrefs.

SMTP_Body (smtp_ID; msgBody{; deleteOption}) \rightarrow Integer

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
msgBody	Text	\rightarrow	Body of message
deleteOption	Integer	\rightarrow	0 = Replace (if msgBody not empty),
			1 = Delete, 2 = Append

Function result Integer ← Error Code

Description

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

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.

Warning: Usually, the body of the message should not contain characters with diacritical marks (such as é, ö, etc.). However, if you want to use such "extended" characters, refer to the SMTP_SetPrefs and SMTP_Charset command descriptions.

deleteOption is an integer value which specifies 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, overriding any prior settings (if you pass an empty string in msgBody, the prior text will be used).
- A value of 1 will set the body to the provided text string, overriding any prior settings (if you pass an empty string in msgBody, the body of the message will be deleted).
- 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*.

deleteOption is an optional parameter which will default to zero if not otherwise specified.

Example

Here is a complete SMTP example:

```
C_LONGINT($SMTP_ID)
C BOOLEAN($SentOK;$OK)
                    `A flag to indicate if we made it through all of the commands
$SentOK:=<u>False</u>
Case of
   : (\underline{\text{Not}}(\mathsf{ERRCHECK}\ ("SMTP\_New"; \underline{\mathit{SMTP\_New}}\ (\$SMTP\_ID))))
     (Not(ERRCHECK ("SMTP_Host"; SMTP_Host ($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 ($$MTP_ID; "X-4Ddemo:"; <>VERSION))))
    : (Not(ERRCHECK ("SMTP_Body"; SMTP_Body ($SMTP_ID; vMessage))))
    : (Not(ERRCHECK ("SMTP_Send"; SMTP_Send ($SMTP_ID))))
Else
   $SentOK:=<u>True</u> `message was composed and mailed successfully
End case
If ($SMTP_ID#0) `If a Message Envelope was created we should clear it now
   $OK:=ERRCHECK ("SMTP_Clear"; SMTP_Clear ($SMTP_ID))
End if
```

Note: For more information about this particular use of the *Case of* structure, please refer to <u>Appendix A, Programming Tips</u>.

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 Charse, SMTP New, SMTP SetPeds.
```

SMTP Attachment (smtp ID; fileName; encodeType{; deleteOption}) → Integer

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

Function result Integer \leftarrow Error Code

Description

The SMTP_Attachment command 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 $\underline{SMTP_New}$ 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.

encode Type 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 <code>encodeType</code> 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 <code>SMTP_Send</code> command is issued. If the file is large it may take some time for the <code>SMTP_Send</code> 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 <code>IT_Encode</code> command and then attach the resulting file to your message using the negative value of <code>encodeType</code>. A negative value in <code>encodeType</code> 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.

Note: You cannot pass an array element to the encodeType parameter.

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



SMTP Charset (encodeHeaders; bodyCharset) -> Integer

Parameter	Type		Description
encodeHeaders	s Integer	\rightarrow	-1 = Use current settings, 0 = Do not manage,
			1 = Convert using the specified charset if ISO-8859-1
			or ISO-2022-JP, encode extended characters
bodyCharset In	Integer	\rightarrow	-1 = Use current settings, 0 = Do not manage,
			1= Convert using the specified charset if ISO-8859-1
			or ISO-2022-JP

Function result Integer \leftarrow Error Code

Description

The SMTP_Charset command allows automatic support of messages containing extended characters while sending them with the SMTP_QuickSend or SMTP_Send commands. If this command is not called or has parameters set to 0, version 6.7 or higher of 4D Internet Commands will work the same way as version 6.5.x.

The <u>SMTP_Charset</u> command allows first, to define if the <u>SMTP_SetPrefs</u> bodyType parameter value must be applied to convert the message headers and body, and second, to define if a header containing extended characters must be encoded using the "=?ISO-8859-1?Q?Test=E9?= ..." syntax as specified in the RFC#1342. This command has an interprocess scope and will have effect on all subsequent messages sent using the <u>SMTP_QuickSend</u> and <u>SMTP_Send</u> in any 4D process.

This command is particularly useful for supporting extended characters included in the message headers such as Subject or mail addresses (for example, address encoding such as "=?ISO-8859-1?Q? Test=E9?= <test@n.net >").

Depending on the message headers, encoding (always set to Base64 (except for Subject header) and depending on the <u>SMTP_SetPrefs</u> bodyType parameter value) will be managed as follows:

- Subject, Comment (called "unstructured headers"): the full string is encoded if it includes extended characters.
- From, To, CC, Bcc, Sender, ReplyTo, InReplyTo (called "structured headers"):
 - Any text between angle brackets ("<", ">") is systematically considered as an e-mail address and is not encoded.
 - Special and delimiter characters such as SPC <> () @ , ; : " / ? . = are not encoded.
 - Strings delimited by special and delimiter characters are encoded if they include extended characters.

Address examples:

- someone@somewhere is not encoded;
- Michèle <michele@somewhere>, only Michèle is encoded.

The encodeHeaders parameter specifies how to handle header conversion and encoding while sending a message. Default value is set to 0.

- -1: Use current settings;
- 0: Do not manage;
- 1: On one hand, if the SMTP_SetPrefs bodyType parameter sets the

character set to ISO-8859-1

or ISO-2022-JP, headers are converted using the specified character set.

- On the other hand, headers are encoded using the following syntax "=? Charset specified

by the SMTP_SetPrefs command? Base64 Encoding?Test=E9?= ..." (see RFC#1342) if they

include extended characters, whatever the specified character set.

- Exception: the Subject header will be encoded using the encoding method given by the

bodyType of the SMTP_SetPrefs command if needed.

Note: Extended headers such as "X_..." must use exclusively US ASCII codes.

The bodyCharset parameter specifies how to handle the message body character set conversion while sending a message. Default value is set to 0.

- · -1: Use current settings;
- · 0: Do not manage;
- 1: if the SMTP_SetPrefs bodyType parameter sets the character set to ISO-8859-1 or ISO-2022-JP, the message body is converted using the specified character set.

Example

In this example, the subject and the body are converted using the ISO-8859-1 character set, and the subject is encoded following the RFC 1342 syntax:

```
SMTP_SetPrefs(1;1;0)
$err:=SMTP_Charset(1;1)
$err:=SMTP_QuickSend("mymail.com";"myaddress";"destination";"the Euro ¤";"the Euro symbo
```

See Also

POP3_Charset, SMTP_SetPrefs.

SMTP_Auth (smtp_ID; userName; password{; authMode}) \rightarrow Integer

Parameter	Type		Description
smtp_ID	Longint	\rightarrow	Message reference
userName	String	\rightarrow	User name to be used for SMTP authentication
password	String	\rightarrow	Password to be used for SMTP authentication
authMode	Integer	\rightarrow	Authentication mode to be used:
			0 or omitted = Mode defined by server
			1 = PLAIN, 2 = LOGIN, 3 = CRAM-MD5

Function result Integer \leftarrow Error code

Description

The SMTP_Auth command allows sending a message referenced by smtp_ID when an authentication mechanism is required by the SMTP server. This type of authentication is required by some SMTP servers in order to reduce the risk that messages have been falsified or that the sender's identity has been usurped, in particular for the purpose of spamming.

This command can be used whether authentication is needed or not since it is only executed if userName and password are not null strings.

 $smtp_ID$ is the long integer reference to the mail message created with the $\underline{SMTP_New}$ command.

userName is the authentication user name on the SMTP server. userName should not contain the domain. For example, for the address "jack@4d.com," userName would just be "jack." password is the authentication password for userName on the SMTP server.

Note: If userName and/or password are null strings, the SMTP_Auth command is not executed.

The optional authMode parameter allows the "forcing" of the authentication mode used. You can pass 0, 1, 2 or 3 in this parameter:

- If you pass 0 (zero), the authentication mode used by the SMTP_Auth command will be the most secure mode supported by the server (CRAM-MD5, LOGIN then PLAIN),
- \bullet If you pass 1, the authentication method used will be PLAIN,
- If you pass 2, the authentication method used will be LOGIN,
- If you pass 3, the authentication method used will be CRAM-MD5.

If authMode is omitted, the value 0 is used by default. If the authentication method requested by this parameter is not supported by the SMTP server, an error is returned.

Example

This example enables sending a message with or without authentication depending on the content of specific fields stored in the 4D database:

```
C_INTEGER($vError)
C_LONGINT($vSmtp_id)
C STRING(30;$vAuthUserName;$vAuthPassword)
$vError:=\frac{SMTP_New}{} (\$vSmtp_id)
$vError:=\frac{SMTP_Host}{(\$vSmtp_id;\"wkrp.com")}
$vError:=\frac{SMTP_From}{\$vSmtp_id;\"herb_tarlick@wkrp.com"}
$vError:=\frac{SMTP_Subject}{\text{Smtp_id;"Are you there?"}}
$vError:=\frac{SMTP_To}{(\$vSmtp_id; "Dupont@wkrp.com")}
$vError:=SMTP_Body($vSmtp_id;"Can we have a meeting?")
     The fields are entered if the server uses an authentication
     mechanism. Otherwise, null strings are returned.
$vAuthUserName:=[Account]AuthUser
$vAuthPassword:=[Account]AuthPass
$vError:=SMTP_Auth($vSmtp_id;$vAuthUserName;$vAuthPassword)
$vError:=\frac{SMTP_Send}{\( \$vSmtp_id\) }
$vError:=SMTP_Clear($vSmtp_id)
```



IC POP3 Review Mail Receiving Mail, Overview

POP3 SetPrefs (stripLineFeed; msgFolder{; attachFolder}) → Integer

POP3 GetPrefs (stripLineFeed; msgFolder{; attachFolder}) → Integer

POP3 Login (hostName; userName; password; aPOP; pop3_ID) → Integer

POP3 Login (hostName; userName; password; aPOP; pop3_ID) → Integer

POP3 Logout (pop3_ID) → Integer

POP3 Reset (pop3_ID) → Integer

POP3 Delete (pop3_ID; startMsg; endMsg) → Integer

POP3 BoxInfo (pop3_ID; msgCount; msgSize) → Integer

POP3 MsgInfo (pop3_ID; msgNumber; msgSize; uniqueID) → Integer

POP3 MsgLstInfo (pop3_ID; msgNumber; msgSize; uniqueID) → Integer

POP3 MsgLstInfo (pop3_ID; msgNumber; msgSize; uniqueID) → Integer

POP3 MsgLstInfo (pop3_ID; startMsg; endMsg; sizeArray; msgNumArray; idArray) → Integer

POP3 MsgLst (pop3_ID; start; end; hdrArray; msgNumArray; idArray; valueArray) → Integer

POP3 DownLoad (pop3_ID; uniqueID; msgNumber) → Integer

POP3_UIDToNum (pop3_ID; uniqueID; msgNumber) → Integer

POP3_Charset (decodeHeaders; bodyCharset) → Integer Other related commands: MSG_Extract (fileName; decode; attachmentPath; enclosureList) Integer -- Theme: IC Downloaded Mail • <u>MSG_SetPrefs</u> (stripLineFeed; msgFolder; attachFolder) Integer -- Theme: IC Downloaded Mail Integer -- Theme: IC Downloaded Mail • MSG_GetPrefs (stripLineFeed; msgFolder; attachFolder) • MSG_Charset (decodeHeaders; bodyCharset) Integer -- Theme: IC Downloaded Mail • <u>SMTP_Charset</u> (encodeHeaders; bodyCharset) Integer -- Theme: IC Send Mail

Receiving Mail, Overview

version 2003 (Modified)

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, "IC POP3 Review Mail" and "IC 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 "IC POP3 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 "IC Downloaded Mail" 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 <u>POP3_Logout</u> 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	bd573a4dbd57	3a4d 1 Jul 1998 	3 jimw@acme.com	Sales lead
2	bd574dc7bd57	4dc7 1 Jul 1998 	frank@acme.com	Site-License order
3	bd575f06bd575	5f06 3 Jul 1998 	ioe@acme.com	Lunch anyone?
4	bd5761d4bd57	61d4 ⁴ Jul 1998 	kelly@acme.com	Your wife called
5	bd577dc7db57	7dc5 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	jimw@acme.com	Sales lead
2	bd574dc7bd574dc7	1 Jul 1998	frank@acme.com	Site-License order
3	bd577dc7db577dc5	4 Jul 1995	track@fedex.com	FedEx tracking

msgNumber 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

Note: For greater flexibility, 4D Internet commands let you pass a POP3, IMAP or FTP connection reference directly to low-level TCP commands and vice versa. For more information, refer to the <u>Low Level Routines</u>, <u>Overview</u> section.



 $POP3_SetPrefs \ (stripLineFeed; msgFolder\{; attachFolder\}) \longrightarrow Integer$

Parameter	Type		Description
stripLineFeed	Integer	\rightarrow	0 = Don't Strip LineFeeds, 1 = Strip LineFeeds,
			-1 = No Change
msgFolder	Text	\rightarrow	Messages folder path ("" = no change)
attachFolder	Text	\rightarrow	Attachments folder path ("" = no change)

Function result Integer ← Error Code

Description

The POP3_SetPrefs command 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.

Compatibility note (version 6.8.1): the stripLineFeed and msgFolder parameters were previously applied to MSG_Commands. This is no longer the case when the MSG_SetPrefs command is used.

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.

Compatibility note (version 6.8.1): the attachFolder parameter is also found in POP3_SetPrefs and MSG_SetPrefs therefore you can modify it using either of these two commands. Using the MSG_SetPrefs command is strongly recommended; the POP3_SetPrefs parameter, used for compatibility reasons, will not be used in the future. The attachFolder of the POP3_SetPrefs command is optional therefore we recommend that you do not pass this parameter. This recommendation also applies to POP3_GetPrefs.

See Also

MSG_Extract, MSG_GetPrefs, MSG_SetPrefs, POP3_DownLoad, POP3_GetPrefs.

POP3_GetPrefs

version 6.8.1 (Modified)

 $POP3_GetPrefs\ (stripLineFeed;\ msgFolder\{;\ attachFolder\}) {\:\longrightarrow\:} Integer$

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

Function result Integer \leftarrow Error Code

Description

The POP3_GetPrefs command 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.

 $attach Folder \ is \ a \ text \ variable \ which \ returns \ the \ local \ pathname \ to \ the \ default \ folder \ in \ which \ extracted \ attachments \ are \ stored.$

Compatibility note (version 6.8.1): The attachFolder parameter of the POP3_GetPrefs command is optional; therefore, we recommend that you do not pass this parameter since it will no longer be used. Note that this parameter does not affect POP3 commands since it is only used by MSG commands.

See Also

MSG_SetPrefs, POP3_SetPrefs.

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

Parameter	Type		Description
hostName	String	\rightarrow	Host Name or IP address of the POP3 mail
	Jung		server
userName	String	\rightarrow	User name
password	String	\rightarrow	Password
aPOP	Integer	\rightarrow	0 = Cleartext Login, 1 = APOP Login
pop3_ID	Longint	\leftarrow	Reference to this POP3 login

Function result Integer \leftarrow Error Code

Description

The <u>POP3_Login</u> command 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 <u>POP3_Loqout</u> 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.

 $userName \ is the \ user's \ name \ on the \ POP3 \ mail \ server. The \ userName \ sould \ not \ contain \ the \ domain. For \ example, for the \ address \ "jack@4d.com", \\ userName \ would \ be just \ "jack".$

 $\it password$ is the password for $\it userName$ 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

version 6.5

POP3_VerifyID (pop3_ID) \longrightarrow Integer

Parameter	Type	
pop3_ID	Longint	

Description

- → Reference to a POP3 login
- \leftarrow 0 = Connection has already closed

Function result Integer \leftarrow 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 <u>POP3_VerifyID</u> 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, <u>POP3_VerifyID</u> will return the appropriate error and free memory used by the POP3 session, and return a zero value back to <u>pop3_ID</u>.

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

See Also

POP3_Login.

version 6.5

POP3_Reset (pop3_ID) \rightarrow Integer

ParameterTypeDescriptionpop3_IDLongint→ Reference to a POP3 login

Function result Integer \leftarrow Error Code

Description

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

Note: The <u>POP3_Delete</u> 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 (<u>POP3_Logout</u>). pop3_ID is a long integer reference to an open session created with <u>POP3_Login</u>.

See Also

POP3_Delete, POP3_Login.

POP3_Delete

version 6.5

POP3_Delete (pop3_ID; startMsg; endMsg) \longrightarrow Integer

Parameter	Type		Description
pop3_ID	Longint	\rightarrow	Reference to a POP3 login
startMsg	Longint	\rightarrow	Starting message number
endMsg	Longint	\rightarrow	Ending message number

Function result Integer \leftarrow Error Code

Description

Given a range of messages from *startMsg* to *endMsg*, the *POP3_Delete* command will mark each message to be deleted. The act of deleting the message does not occur until you successfully issue the *POP3_Loqout* command. If your current session terminates for any reason (timeout, network failure, etc.) prior to calling the *POP3_Loqout* 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.

 ${\it endMsg} \ {\rm is\ a\ long\ integer\ number\ which\ is\ the\ ending\ message\ number\ of\ the\ messages\ to\ delete}.$

Note: The POP3_Delete, POP3_MsqLstInfo and POP3_MsqLst 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.

version 6.5

POP3_Logout (pop3_ID) \rightarrow Integer

Parameter Type pop3_ID Longint

Description

- → Reference to a POP3 login
- \leftarrow 0 = Command successfully logs off

Function result Integer \leftarrow Error Code

Description

The POP3 Logout command 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 <u>POP3_Login</u>.

See Also

POP3_Reset.

POP3_BoxInfo

version 6.5

POP3_BoxInfo (pop3_ID; msgCount; msgSize) \longrightarrow Integer

Parameter	Type		Description
pop3_ID	Longint	\rightarrow	Reference to a POP3 login
msgCount	Longint	\leftarrow	Number of messages
msgSize	Longint	\leftarrow	Size of all messages

Function result Integer \leftarrow Error Code

Description

The POP3_BoxInfo command 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

version 6.5

POP3_MsgInfo (pop3_ID; msgNumber; msgSize; uniqueID) \rightarrow Integer

Parameter	Type		Description
pop3_ID	Longint	\rightarrow	Reference to a POP3 login
msgNumber	Longint	\rightarrow	Message number
msgSize	Longint	\leftarrow	Message size
uniqueID	String	\leftarrow	Unique ID of message on server

Function result Integer \leftarrow Error Code

Description

The POP3_MsgInfo command 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 <u>POP3_Loqin</u>.

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

version 6.5

POP3_GetMessage (pop3_ID; msgNumber; offset; length; msgText) \longrightarrow Integer

Parameter	Type		Description
pop3_ID	Longint	\rightarrow	Reference to a POP3 login
msgNumber	Longint	\rightarrow	Message number
offset	Longint	\rightarrow	Offset of character at which to begin retrieval
length	Longint	\rightarrow	How many characters to return
msgText	Text	\leftarrow	Message Text

Function result Integer \leftarrow Error Code

Description

The POP3_GetMessage command 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 is limited to 32,000 characters, the *length* parameter should be set to any number below 32,000. Messages whose size is greater than 32K must be retrieved to disk via the *POP3_Download* command.

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

See Alse

POP3_DownLoad, POP3_SetPrefs.

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

Parameter	Type		Description
pop3_ID	Longint	\rightarrow	Reference to a POP3 login
startMsg	Longint	\rightarrow	Start message number
endMsg	Longint	\rightarrow	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 \leftarrow Error Code

Description

The POP3_MsgLstInfo command 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.

 ${\it msgNumArray} \ {\rm is\ a\ long\ integer\ array\ returned\ containing\ the\ message\ numbers\ between\ startMsg\ and\ endMsg.}$

 $id Array \ is \ a \ string \ or \ text \ array \ returned \ containing \ the \ Unique-ID's \ of \ the \ messages \ between \ start Msg \ and \ end Msg.$

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	\rightarrow	Reference to a POP3 login
start	Longint	\rightarrow	Start message number
end	Longint	\rightarrow	End message number
hdrArray	Str Txt Array	\rightarrow	Array of Headers to retrieve
msgNumArray	Longint Array	\leftarrow	Array of message numbers
idArray	String Array	\leftarrow	String array of Unique ID's
valueArray	2D Str Txt Array	\leftarrow	2D Array of header values

Function result Integer \leftarrow Error Code

Description

The POP3_MsgLst command 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.

Note: Since mail headers can include extended characters, you can automate their management using the POP3_Charset command.

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@4d.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_MsqLstInfo 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_Charset, POP3_MsgInfo, POP3_MsgLstInfo.



POP3_DownLoad (pop3_ID; msgNumber; headerOnly; fileName) -> Integer

Type		Description
Longint	\rightarrow	Reference to a POP3 login
Longint	\rightarrow	Message number
Integer	\rightarrow	0 = Entire message, $1 = $ Only header
Text	\rightarrow	Local Filename
	\leftarrow	Resulting Local Filename
	Longint Longint Integer	$\begin{array}{ccc} \text{Longint} & \rightarrow \\ \text{Longint} & \rightarrow \\ \text{Integer} & \rightarrow \\ \text{Text} & \rightarrow \end{array}$

Function result Integer \leftarrow Error Code

Description

The POP3_DownLoad command 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:

1111	= 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 (with 4D single-user) or in the 4D Client folder (with 4D Server). 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_ID; uniqueID; msgNumber) \longrightarrow Integer

Parameter	Type		Description
pop3_ID	Longint	\rightarrow	Reference to a POP3 login
uniqueID	String	\rightarrow	Unique ID of message on server
msgNumber	Longint	\leftarrow	Message number

Function result Integer \leftarrow Error Code

Description

The POP3_UIDToNum command 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 $\underline{POP3_Loqin}$.

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.

POP3_Charset (decodeHeaders; bodyCharset) -> Integer

Parameter Type

decodeHeaders Integer

bodyCharset Integer

Description

- → -1 = Use current settings, 0 = Do not manage,
 1 = Convert using the Mac OS charset if ISO-8859-1
 - or ISO-2022-JP, decode extended characters
- → -1 = Use current settings, 0 = Do not manage,
 1 = Convert using the Mac OS charset if ISO-8859-1
 - or ISO-2022-JP

Function result Integer ← Error Code

Description

The POP3_Charset command allows automatic support of messages containing extended characters while processing them with the POP3 and MSG commands. If this command is not called or has parameters set to 0, version 6.7 or higher of 4D Internet Commands will work the same way as version 6.5.x.

POP3_Charset allows first, to set whether the extended characters header decoding has to be managed, and second, whether the message body and headers character set conversions have to be managed.

This command is particularly usefull to support extended characters included in message headers such as "Subject" or mail addresses (for example, to decode an address such as "=?ISO-8859-1?Q? Test=E9?= < test@n.net >").

The decodeHeaders parameter specifies how to handle header decoding and conversion while executing commands POP3_MsqLst or MSG_FindHeader(see Compatibility note). Default value is set to 0.

- -1: Use current settings;
- 0: Do not manage;
- 1: Headers are decoded if necessary. If decoded and if the specified character set is ISO-8859-1 or ISO-2022-JP, headers are converted using respectively the Mac OS ASCII code or the Shift-JIS.

Compatibility note (version 6.8.1): POP3_Charset applies to MSG_FindHeader (in the same way as POP3_MsgLst) if the MSG_Charset command has not been executed previously.

The bodyCharset parameter specifies how to handle message body character set conversion while executing the MSG_GetBody command (see Compatibility note). Default value is set to 0.

- -1: Use current settings;
- 0: Do not manage;
- 1: If the "Body-Content-Type" character set is set to ISO-8859-1 or ISO-2022-JP, the message body is converted using respectively the Mac OS ASCII code or the Shift-JIS.

Compatibility note (version 6.8.1): POP3_Charset applies to MSG_GetBody if the MSG_Charset command has not been executed previously.

Examples

1. Using version 6.5.x of 4D Internet Commands:

```
$Err:=MSG_FindHeader($msgfile;"From";$from)
   $from:=ISO to Mac($from)
   $Err:=MSG_FindHeader($msgfile;"To";$to)
   $to:=<u>ISO</u> to <u>Mac</u>($to)
   $Err:=MSG_FindHeader($msgfile;"Cc";$cc)
   $cc:=<u>ISO</u> to <u>Mac</u>($cc)
   $Err:=MSG FindHeader($msgfile; "Subject"; $subject)
   $subject:=<u>ISO to Mac(</u>$subject)
   $Err:=MSG_MessageSize($msgfile;$HdrSize;$BdySize;$MsgSize)
   $Err:=MSG GetBody($msgfile;0;$BdySize;$BodyContent)
   $BodyContent:=<u>ISO to Mac(</u>$BodyContent)
2. Using version 6.7 of 4D Internet Commands:
   $Err:=POP3_Charset(1;1)
   $Err:=MSG FindHeader($msgfile;"From";$from)
   $Err:=MSG FindHeader($msgfile;"To";$to)
   $Err:=MSG_FindHeader($msgfile;"Cc";$cc)
   $Err:=MSG_FindHeader($msgfile; "Subject"; $subject)
```

```
$Err:=MSG MessageSize($msgfile;$HdrSize;$BdySize;$MsgSize)
   $Err:=MSG GetBody($msgfile;0;$BdySize;$BodyContent)
3. Using version 6.8 of 4D Internet Commands:
   $Err:=MSG Charset(1;1)
   $Err:=MSG FindHeader($msgfile;"From";$from)
   $Err:=MSG FindHeader($msgfile;"To";$to)
$Err:=MSG FindHeader($msgfile;"Cc";$cc)
   $Err:=MSG_FindHeader($msgfile; "Subject"; $subject)
   $Err:=MSG MessageSize($msgfile;$HdrSize;$BdySize;$MsgSize)
   $Err:=MSG GetBody($msgfile;0;$BdySize;$BodyContent)
MSG_Charset, SMTP_Charset.
```

IC IMAP Review Mail

 IMAP4 Commands, Overview

 IMAP SetPrefs (stripLineFeed; msgFolder) → Integer

 IMAP GetPrefs (stripLineFeed; msgFolder) → Integer

 IMAP Login (hostName; userName; userswing; password; imap_ID) → Integer

 IMAP VerifyID (imap_ID) → Integer

 IMAP Capability (imap_ID); mbRefName; mbName; mbNamesArray; mbAttribsArray; mbHierarArray{; subscribedMB}) → Integer

 IMAP SubscribeMB (imap_ID; mbName; mspNber; newMsgNber; unseenMsgNber; mbUID) → Integer

 IMAP GetMBStatus (imap_ID; mbName; msgNber; newMsgNber; customFlags; permanentFlags; mbUID) → Integer

 IMAP GetCurrentMB (imap_ID; mbName; msgNber; newMsgNber; customFlags; permanentFlags; mbUID) → Integer

 IMAP GetCurrentMB (imap_ID; mbName; msgNber; newMsgNber; customFlags; permanentFlags; mbUID) → Integer

 IMAP Delete (imap_ID; startMsg; endMsg) → Integer

 IMAP MsgInfo (imap_ID; startMsg; endMsg) → Integer

 IMAP MsgLst (imap_ID; startMsg; endMsg; msgSizeArray; msgNumArray; msgIdArray) → Integer

 IMAP MsgLst (imap_ID; startMsg; endMsg; msgFlagsList; deleteOption) → Integer

 IMAP MsgLst (imap_ID; startMsg; endMsg; msgFlagsArray; msgNumArray) → Integer

 IMAP SetFlags (imap_ID; msgNum; headerOnly; fileName{; updateSeen}) → Integer

 IMAP Download (imap_ID; msgNum; msgDataltem; msgDataltemValue) → Integer

 IMAP Download (imap_ID; msgNum; headerOnly; fileName{; updateSeen}) → Integer

 IMAP MsgNumToUID (imap_ID; msgNum; headerOnly; fileName{; updateSeen}) → Integer<

IMAP4 Commands, Overview

version 2003 (Modified)

The set of IMAP commands enables your database to access and manipulate electronic mail messages over an IMAP electronic mail server and to retrieve electronic messages from your IMAP server. IMAP commands are compliant with the Internet Message Access Protocol, Version 4 revision 1 (IMAP4rev1), defined by rfc 2060. IMAP4rev1 allows the managing of remote message folders, called "mailboxes", in a way that is functionally similar to local mailboxes.

IMAP commands include operations to create, delete and rename mailboxes; check for new messages; permanently remove messages; set and clear message flags; search messages; and retrieve selective message parts.

Terminology

"Connection" refers to the entire sequence of IMAP client/server interaction from the initial network connection (IMAP_Login) until the end of the selection (IMAP_Logout).

"Session" refers to the sequence of client/server interaction from the moment a mailbox is selected (<u>IMAP_SetCurrentMB</u>) until the end of the selection (<u>IMAP_SetCurrentMB</u>, <u>IMAP_CloseCurrentMB</u>) or until the connection is closed (<u>IMAP_Logout</u>).

IMAP Connection Overview

- Initializing TCP communication: IT_MacTCPInit (the IT_PPPConnect command must be called before IT_MacTCPInit in the event of a PPP connection).
- · Opening a connection: IMAP_Login
- $\bullet \ Managing \ mailboxes: List, create, delete, rename, subscribe/unsubscribe, and/or \ get \ status \ parameters.$
- Opening a session by defining the current working mailbox: IMAP_SetCurrentMB.

Once the current mailbox is set, you can manage messages for it.

- Managing messages: List, download or delete messages; list message flags; modify message flags; copy to another mailbox; search and retrieve e-mail parts without any downloading, etc.
- Once you are finished working with the current mailbox messages, you can close the session or open a new one by setting another current mailbox. In any case, the IMAP server will permanently update its messages. For instance, it will delete all messages with the **'Deleted** flag set.
- Once you are finished, you should logout. Closing a connection: IMAP_Logout.
- · Other operations: Preference settings, capability, check connection, and reset any inactivity auto-logout timer on the IMAP server.

IMAP Command Themes

The IMAP-related commands are divided into two sections: IC IMAP Review Mail and IC Downloaded Mail. These commands have been separated to show the different methods of reading electronic mail. When reading electronic mail from an IMAP server, messages (or message information) may be imported into 4D structures (variables, fields, arrays) or downloaded to the disk. This section details the 4D Internet Commands' capacity to read messages from an IMAP server.

The need for dual message-retrieval methods is due mainly to memory constraints for actions that may download large amounts of information. For instance, a single message containing a 5-MB attachment could easily overflow the database's storage capacity. A picture or BLOB field is the only 4D structure capable of storing something of this size; however, converting a message or attachment to this format is not very efficient since accessing the picture or BLOB involves cumbersome memory requirements for the client. To resolve this problem, this section has an MAP_Download command which transfers messages from the IMAP server to the user's hard disk.

Once imported to the disk, the "IC Downloaded Mail" section details the commands used to manipulate local files.

Mailbox mechanisms

An IMAP mailbox can be handled like a folder and may contain files and/or subfolders. Similarly, a mailbox may contain messages and/or submailboxes.

A mailbox is accessed using its complete hierarchical name. Depending on the IMAP server, each hierarchical level is separated by a hierarchy separator (a separator is returned using the IMAP_ListMBs command).

You can use the separator to create child mailboxes and to search higher or lower levels of the naming hierarchy. All children of a top-level hierarchy node use the same separator character.

Note: Messages can only be managed once the current working mailbox has been selected (IMAP_SetCurrentMB).

Each account can have one or several mailboxes.

Mailbox names are still case-sensitive; therefore, you cannot create two mailboxes with names that differ only in case.

The INBOX mailbox is a particular case: it exists in every account and is used for storing incoming messages. The INBOX is created automatically whenever an account is set up.

A user cannot remove the INBOX mailbox but may rename it. If he chooses to rename it, a new empty INBOX is immediately created. The INBOX name is never case-sensitive.

Some mailbox attributes, such as the total number of messages or new messages, may be checked even if the mailbox is not the current one.

msgNum and uniqueID

When using IMAP commands, it is important to fully understand the most frequently used parameters; more particularly, mailbox mechanisms msqNum and uniqueID.

msgNum is the number of a message in the mailbox at the time the $\underline{IMAP_SetCurrentMB}$ command is executed.

Once a current mailbox is selected, messages in the mailbox are assigned numbers starting from 1 up to the total number of items in the mailbox. Numbers are assigned based on the order that the messages were received in the mailbox with 1 being the oldest. The numbers assigned to messages are only valid from the moment you select the current working mailbox (<u>IMAP_SetCurrentMB</u>) until it is closed (<u>IMAP_CloseCurrentMB</u>, <u>IMAP_SetCurrentMB</u> or <u>IMAP_Loqout</u>).

When the mailbox is closed, all messages marked for deletion will be removed.

When the user logs back onto the IMAP server, the current messages in the mailbox will once again be numbered from 1 to X. For instance, if there are 10 messages in the mailbox, and messages 1 to 5 are deleted, when the user reopens the mailbox, the former messages 6 to 10 will have been re-numbered 1 to 5.

For instance, consider the example below: You log on to an IMAP server and obtain the following list of messages:

msgNum	uniqueID	Date	From	Subject
1	10005	1 Jul 2001	danw@acme.com	Sales lead
2	10008	1 Jul 2001	frank@acme.com	Site-License order

3	10012	3 Jul 2001	joe@acme.com	Lunch anyone?
4	20000	4 Jul 2002	kelly@acme.com	Your wife called
5	20001	4 Jul 2002	track@fedex.com	FedEx tracking

During this session, you delete messages number 3 and 4. When you close the current working mailbox, the requested deletions are made. When you log back onto the server, your message list will be renumbered as follows:

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

msgNum is not a static value and will vary from one session to another. It will change in relation to other messages in the mailbox at the time the current working mailbox is selected.

However, the *uniqueID* is a unique number, assigned to the message by the IMAP server in a strictly ascending order. As each message is added to the mailbox, it is assigned a higher ID than the previously added message(s).

Unfortunately, IMAP servers do not use the uniqueID as the primary reference for their messages. When using IMAP commands you will need to specify the msgNum as the reference for messages on the server. Developers may need to be careful when developing solutions which import message references into a database, while leaving the message body itself on the server.

Recommendations

Since the whole point of IMAP is interoperability, and since the latter cannot be tested in a vacuum, the final recommendation is "Test for EVERYTHING." Therefore, test your client against every server you can get an account on.

For more information, please check out the following sites:

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

POP3 and IMAP4 commands comparison

Login VerifyID	Exactly equiv Exactly equiv	No POP parameter for IMAP
Delete	Exactly equiv	IMAP commands delete in real time. POP3 waits for
		POP3_Logout to remove messages permanently.
		IMAP_SetFlags with \Deleted flag allows you to
		obtain the same result as the POP3_Delete command
Logout	Exactly equiv	
SetPrefs	Exactly equiv	No attachFolder for IMAP,
		POP3 attachFolder has become optional
GetPrefs	Exactly equiv	See attachFolder note in SetPrefs
MsgLstInfo	Exactly equiv	
MsgInfo	Exactly equiv	
MsgLst	Exactly equiv	

Download Exactly equiv No direct equiv Need combination of IMAP_Search on \Deleted flags and IMAP_SetFlags to remove the \Deleted flag POP3_BoxInfo No direct equiv No direct equiv No direct Need combination of IMAP_SetCurrentMB **EMAP_MsgLstInfo commands** No direct No direct No direct	UIDToMsgNum	Exactly equiv	IMAP msgUID is a Longint, POP3 msgUID is a string
POP3_Reset equiv Need combination of IMAP_Search on \Deleted flags and IMAP_SetFlags to remove the \Deleted flag No direct equiv No direct equiv No direct Need combination of IMAP_Search on remove the \Deleted flag Need combination of IMAP_SetFlags to remove the \Deleted flag Need combination of IMAP_SetFlags to remove the \Deleted flag Need combination of IMAP_Search on remove the \Deleted flag Need combination of IMAP_SetFlags to remove the \Deleted flag Need combination of IMAP_SetFlags to remove the \Deleted flag Need combination of IMAP_SetFlags to remove the \Deleted flag Need combination of IMAP_SetFlags to remove the \Deleted flag Need combination of IMAP_SetFlags to remove the \Deleted flag Need combination of IMAP_SetFlags to remove the \Deleted flag Need combination of IMAP_SetFlags to remove the \Deleted flag Need combination of IMAP_SetFlags to remove the \Deleted flag Need combination of IMAP_SetFlags to remove the \Deleted flags Need combination of IMAP_SetFlags to remove the \Deleted flags Need combination of IMAP_SetFlags to remove the \Deleted flags Need combination of IMAP_SetFlags to remove the \Deleted flags Need combination of IMAP_SetFlags to remove the \Deleted flags Need combination of IMAP_SetFlags to remove the \Deleted flags Need combination of IMAP_SetFlags to remove the \Deleted flags Need combination of IMAP_SetFlags to remove the \Deleted flags and IMAP_SetFlags to remove the \Deleted flags to \Delete	Download	Exactly equiv	
POP3_BoxInfo No direct equiv Red combination of IMAP_SetCurrentMB &IMAP_MsgLstInfo commands No direct	POP3_Reset		Need combination of IMAP_Search on
POP3_BoxInfo No direct equiv Need combination of IMAP_SetCurrentMB &IMAP_MsgLstInfo commands No direct			\Deleted flags and IMAP_SetFlags to
equiv IMAP_SetCurrentMB &IMAP_MsgLstInfo commands			remove the \Deleted flag
equiv IMAP_SetCurrentMB &IMAP_MsgLstInfo commands No direct	POP3 BoxInfo		
No direct	1 01 0 <u>_</u> D0/////	equiv	
IMAD McgNumToLIID No direct			&IMAP_MsgLstInfo commands
INIAP MISGINUITIOOID .	IMAP_ MsgNumToUID		
equiv	_ 0	equiv	
GetMessage Almost Equiv IMAP is more powerful since it allows you	GetMessage	Almost Equiv	•
to choose one additional msgPart which is			•
"only body"			"only body"
POP3_Charset No Equiv IMAP automatically manages charset	POP3_Charset	No Equiv	IMAP automatically manages charset
IMAP_Capability No Equiv Specific to IMAP protocol	IMAP_Capability	No Equiv	Specific to IMAP protocol
IMAP_ListMBs No Equiv Specific to IMAP protocol	IMAP_ListMBs	No Equiv	Specific to IMAP protocol
IMAP_GetMBStatus No Equiv Specific to IMAP protocol	IMAP_GetMBStatus	No Equiv	Specific to IMAP protocol
IMAP_SetCurrentMB No Equiv Specific to IMAP protocol	IMAP_SetCurrentMB	No Equiv	Specific to IMAP protocol
IMAP_GetCurrentMB No Equiv Specific to IMAP protocol	IMAP_GetCurrentMB	No Equiv	Specific to IMAP protocol
IMAP_CloseCurrentMB No Equiv Specific to IMAP protocol	IMAP_CloseCurrentME	3 No Equiv	Specific to IMAP protocol
IMAP_CopyToMB No Equiv Specific to IMAP protocol	IMAP_CopyToMB	No Equiv	Specific to IMAP protocol
IMAP_SubscribeMB No Equiv Specific to IMAP protocol	IMAP_SubscribeMB	No Equiv	Specific to IMAP protocol
IMAP_CreateMB No Equiv Specific to IMAP protocol	IMAP_CreateMB	No Equiv	Specific to IMAP protocol
IMAP_DeleteMB No Equiv Specific to IMAP protocol	IMAP_DeleteMB	No Equiv	Specific to IMAP protocol
IMAP_RenameMB No Equiv Specific to IMAP protocol	IMAP_RenameMB	No Equiv	Specific to IMAP protocol
IMAP_SetFlags No Equiv Specific to IMAP protocol	IMAP_SetFlags	No Equiv	Specific to IMAP protocol
IMAP_GetFlags No Equiv Specific to IMAP protocol	IMAP_GetFlags	No Equiv	Specific to IMAP protocol
IMAP_Search No Equiv Specific to IMAP protocol	IMAP_Search	No Equiv	Specific to IMAP protocol
IMAP_MsgFetch No Equiv Specific to IMAP protocol	IMAP_MsgFetch	No Equiv	Specific to IMAP protocol

Notes

• IMAP and POP3 servers: in the	he case of the IMAP server, do not typ	pe msgID the same way since	msgID is a Long Integer.		
• Deletion does not work in exa the \ Deleted flag; to get the san	actly the same way between POP3 and ne result as <u>POP3_Reset</u> , use the <u>IMA</u>	d IMAP protocols. <u>IMAP_De</u> <u>AP_SetFlags</u> to retrieve the \D	ete removes messages in real tireleted flags.	ne. To get the same result as <u>POP3</u>	Delete, use the IMAP_SetFlags to s
• For greater flexibility, 4D Inte Routines, Overview section	ernet commands let you pass a POP3,	IMAP or FTP connection ref	erence directly to low-level TCF	commands and vice versa. For mo	re information, refer to the <u>Low Leve</u>

 $IMAP_SetPrefs \ (stripLineFeed; msgFolder) \longrightarrow Integer$

Parameter	Type
-----------	------

stripLineFeed Integer

Description

→ 0 = Do not strip LineFeeds, 1 = Strip LineFeeds,

-1 = No Change

→ Messages folder path ("" = no change)

Function result Integer ← Error code

msgFolder

Description

The IMAP_SetPrefs command sets the preferences for all IMAP commands.

Text

stripLineFeed is an integer value specifying how LineFeed characters will be treated in saved messages. Most IMAP 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 IMAP 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 is set to 1 and will automatically strip linefeeds found in messages.

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

See Also

IMAP_Download, IMAP_GetPrefs.

 $IMAP_GetPrefs \ (stripLineFeed; msgFolder) \longrightarrow Integer$

Parameter Type

stripLineFeed Integer

Description

0 = Do not strip LineFeeds, 1 = Strip LineFeeds,

-1 = No Change

Messages folder path

Function result Integer ← Error code

msgFolder

Description

The ${\it IMAP_GetPrefs}$ command returns the current preferences for the IMAP commands.

Text

The preferences are returned into the variables listed in the parameters.

stripLineFeed returns the current setting of the user's preference for linefeed stripping.

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

See Also

IMAP_SetPrefs.

IMAP_Login (hostName; userName; password; imap_ID) → Integer

Parameter	Type		Description
hostName	String	\rightarrow	Host Name or IP address of the IMAP server
userName	String	\rightarrow	User name
password	String	\rightarrow	Password
imap_ID	Longint	\leftarrow	Reference to this IMAP login

Function result Integer ← Error code

Description

The IMAP_Login command logs the user onto the IMAP electronic mail server with the given user name and password.

This particular login is given a connection reference ($\mathit{imap_ID}$) to which subsequent IMAP commands can refer.

The connection is closed using the <u>IMAP_Loqout</u> command or when the IMAP server inactivity timer has timed out.

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

userName is the user's name on the IMAP electronic mail server. The userName should not contain the domain. For example, for the address "jack@4d.com", the userName would be just "jack".

 $\it password$ is the password for the $\it userName$ on the IMAP electronic mail server.

imap_ID is a long integer variable into which a reference to the connection just established is returned. This parameter must be passed a 4D variable in order to accept the returned results. The variable will be used in all subsequent commands which perform actions related to this session.

If IMAP_Login fails, imap_ID is set to zero.

Example

Here is a typical connection sequence:

```
$ErrorNum:=IMAP_Login (vHost;vUserName;vUserPassword;vImap_ID)
If($ErrorNum =0)
    C_TEXT(vCapability)
$ErrorNum:=IMAP_Capability (vImap_ID;vCapability))
... ` IMAP commands using vImap_ID parameter
End if
$ErrorNum:=IMAP_Logout (vImap_ID)
```

See Also

IMAP_Logout, IMAP_VerifyID.

version 6.8.1

 $IMAP_VerifyID \ (imap_ID) {\:\longrightarrow\:} Integer$

Parameter Type imap_ID Longint

Description

- → Reference to this IMAP login
- \leftarrow 0 = Connection has already closed

Function result Integer \leftarrow Error code

Description

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

When executed, the <u>IMAP_VerifyID</u> command will verify the connection has not already been closed. If the connection is still opened, the command will tell the IMAP server to reset the timeout counter for the connection back to zero. If the connection has already closed, <u>IMAP_VerifyID</u> will return the appropriate error and free memory used by the IMAP connection, and return a zero value back to <u>imap_ID</u>.

imap_ID is a long integer reference to an open connection created with IMAP_Login.

See Also

IMAP_Login.

 $IMAP_Capability\ (imap_ID;\ capability) \longrightarrow Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to this IMAP login
capability	Text	\leftarrow	IMAP capabilities

Function result Integer \leftarrow Error code

Description

The IMAP_Capability command returns a text area containing a space-separated listing of capability names supported by the IMAP server. This list determines what version of IMAP, and what optional features (such as extension, revision or amendment to the IMAP4rev1 protocol), a server supports.

 ${\it IMAP4rev1} \ {\it must appear} \ {\it in the capability text to ensure compliance with 4D \ Internet \ Commands}.$

See Also

IMAP_Login.

 $IMAP_ListMBs\ (imap_ID;\ mbRefName;\ mbName;\ mbNamesArray;\ mbAttribsArray;\ mbHierarArray\{;\ subscribedMB\}) \xrightarrow{} Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
mbRefName	Text	\rightarrow	Null string or Mailbox name or
			level of mailbox hierarchy
mbName	Text	\rightarrow	Null string or MailBoxName or wildcards
mbNamesArray	Str Txt Array	←	Array of mailbox names (pathnames)
mbAttribsArray	Str Txt Array	←	Array of mailbox attributes
mbHierarArray	Str Txt Array	\leftarrow	Array of hierarchy delimiters
subscribedMB	Integer	\rightarrow	0 = List all available user mailboxes
			1 = List only subscribed mailboxes

Function result Integer ← Error code

Description

The IMAP_ListMBs command returns the list of available mailboxes for the connected user and attached information. If this command fails, specified arrays are initialized.

mbRefName and mbName must be considered together since the resulting Mailbox list will depend on the combination of these two parameter values.

The returned list may be restricted to subscribed mailboxes (see <u>IMAP_SubscribeMB</u>) when the last parameter, subscribedMB, is set to 1.

When the execution of *IMAP_ListMBs* is very long, either because a large number of mailboxes are being scanned, or because of numerous and complex hierarchical mailbox structures, and so on, you can:

- use wildcards (see below) with IMAP_ListMBs,
- or use the IMAP_ListMBs command, with the subscribedMB parameter set to 1, to list only a set of mailboxes defined using the IMAP_SubscribeMB command.

imap_ID is a long integer reference to an open connection created with IMAP_Login.

mbRefName is a text value that should be combined with the mbName parameter to determine which mailboxes should be looked for. The reference name (mbRefName) should be used as a Current Working Directory on Unix systems. In other words, the mailbox name (mbName) is interpreted as a file located in the directory specified by the reference name (mbRefName). Be sure that the IMAP specification leaves the interpretation of the reference name (mbRefName) "implementation-dependent". We strongly recommend that the user be given an operational mode that does not use any mbRefName reference argument. As such, it can interoperate with older servers that did not implement the use of reference arguments.

If mbRefName is a null string, only the mbName parameter is used to list mailboxes.

If mbRefName contains the name of a mailbox or a level of mailbox hierarchy, it should be used to define the context in which the mbName parameter must be interpreted.

Note: We strongly recommend that you place a trailing hierarchy delimiter on the reference argument when used. This will ensure full compliance whichever IMAP server is used.

mbName is a text value, intended to be combined with the mbRefName parameter, which usually defines the context in which the mbName parameter must be interpreted.

If *mbName* is a null string, the hierarchy delimiter is returned.

Note: If you implement a breakout facility using the *mbRefName* parameter, you should allow the user to choose whether or not to use a leading hierarchy delimiter on the mailbox argument. This is because the handling of a leading mailbox hierarchy delimiter varies from one server to another, and even between different electronic mail stores on the same server. In some cases, a leading hierarchy delimiter means "discard the reference argument"; whereas in other cases, the two are concatenated and the extra hierarchy delimiter is discarded.

The mbNamesArray array receives the list of available mailboxes' names

The mbAttribsArray array receives the list of available mailboxes' attributes.

Mailbox attributes

There are four mailbox attributes defined as follows:

- \Noinferiors: no child levels currently exist and none can be created.
- \Noselect: this name cannot be used as a selectable mailbox.
- •\Marked: the server has marked the mailbox as "interesting"; the mailbox probably contains messages added since the last selection.
- \Unmarked: the mailbox does not contain any additional messages since the last selection.

The mbHierarArray array receives the list of available mailboxes' hierarchy delimiters.

The hierarchy delimiter is a character used to delimit hierarchy levels in a mailbox name. You can use it to create child mailboxes and to search higher or lower levels of the naming hierarchy. All children of a top-level hierarchy node use the same separator character.

subscribedMB is an integer value which can be specified when you simply want to list "subscribed" mailboxes. A zero value lists all available user mailboxes. A value of 1 only lists subscribed user mailboxes. subscribedMB is an optional parameter that will default to zero if not otherwise specified.

Examples

1. The following example:

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

... returns all available mailboxes from the "4DIC/Work/Test" mailbox.

Remember that if the IMAP server does not interpret as was intended, do not use the mbRefName and concatenate the mbRefName and mbName values into mbName:

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

2. The following example:

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

... returns the hierarchy delimiter.

Using the Wildcard character

You can use wildcards in the mbRefName and mbName parameters to make mailbox selection easier. You will find an example of current wildcards below, but please note that the interpretation of wildcards will depend on the IMAP server; consequently, these examples may not work. In this case, check your IMAP server wildcards.

• " * " matches zero or more characters in its position:

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

... returns all mailboxes available to the connected user.

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

... returns all available mailboxes matching the root "Work".

• "%" is similar to "*", but it does not match a hierarchy delimiter. If the "%" wildcard is the last character of the mbName parameter, matching hierarchy levels are also returned. If these hierarchy levels are not selectable mailboxes, they are returned with the \Noselect mailbox attribute (see paragraph "Mailbox attributes").

... returns all mailboxes matching the root "Work", plus one hierarchy level available for the connected user.

"%" can be helpful in order to parse the mailbox hierarchy level by level.

Given the following mailbox hierarchy:

INBOX

MailboxA

MailboxAA

MailboxAB

MailboxB

MailboxBA

MailboxBB

MailboxC

MailboxCA MailboxCB

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

... returns INBOX, MailboxA, MailboxB and MailboxC.

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

... returns MailboxAA and MailboxAB.

Using this technique, you can give the user complete flexibility without being bogged down by the voluminous reply to IMAP_ListMBs(imap_ID;"":"*";mbNamesArray;mbAttribsArray;mbHierarArray).

Note that IMAP servers themselves may limit the number of levels to be scanned.

See Also



IMAP_SubscribeMB

version 6.8.1

 $IMAP_SubscribeMB\ (imap_ID;\ mbName;\ mbSubscribe) \longrightarrow Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
mbName	Text	\rightarrow	Name of the mailbox to subscribe or unsubscribe
mbSubscribe	Integer	\rightarrow	0 = Do not subscribe; 1= Subscribe

Function result Integer \leftarrow Error code

Description

 $The {\it IMAP_SubscribeMB} command allows adding or removing of the specified mailbox name to/from the IMAP server's set of "subscribed" user mailboxes.$

As such, the user can choose to narrow down a large list of available mailboxes by subscribing to those that he usually wants to see. To do this, he simply has to use the <u>IMAP_ListMBs</u> command with the subscribedMB optional parameter set to 1 (see <u>IMAP_ListMBs</u>).

 $imap_ID$ is a long integer reference to an open connection created with $\underline{IMAP_Login}$.

mbName is the full name of the mailbox to be subscribed or unsubscribed.

Pass 0 in mbSubscribe to not subscribe to the mailbox; pass 1 to subscribe.

See Also

IMAP_ListMBs.

 $IMAP_GetMBStatus\ (imap_ID;\ mbName;\ msgNber;\ newMsgNber;\ unseenMsgNber;\ mbUID) \longrightarrow Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
mbName	Text	\rightarrow	Name of the mailbox
msgNber	Longint	\leftarrow	Number of messages in the specified mailbox
newMsgNber	Longint	\leftarrow	Number of messages with the \Recent flag set
unseenMsgNbe	r Longint	\leftarrow	Number of messages with no \Seen flag
mbUID	Longint	\leftarrow	Specified mailbox unique identifier

Function result Integer \leftarrow Error code

Description

The <code>IMAP_GetMBStatus</code> command returns the status parameter values of the mailbox specified by <code>mbName</code>. It does not change the current mailbox (see <code>IMAP_SetCurrentMB</code>), nor does it affect the state of any messages in the specified mailbox (in particular, it usually does not cause messages to lose the <code>\Recent</code> flag, but this can vary depending on the <code>IMAP4</code> server implementation). This is an alternative used to check mailbox status parameters without deselecting the current mailbox.

This command is particularly useful to:

- · Check or retrieve the mailbox unique identifier, and/or,
- Check recent and unseen messages without opening a session for the mailbox.

Important: We strongly recommend that you do not call the *IMAP_GetMBStatus* command using the current mailbox. By doing so, you may encounter problems and the information returned will not necessarily be synchronized with the current mailbox status (in particular for new e-mails).

 $imap_ID$ is a long integer reference to an open connection created with $\underline{IMAP_Login}$.

mbName is the full name of the existing mailbox for which you want to get the status parameter values.

Note: Unlike the IMAP_ListMBs command, the mbName parameter does not accept wildcards.

msgNber returns the number of messages in the current mailbox (set to zero when the command is called and returns -1 if error).

newMsgNber returns the number of recent messages in the current mailbox (set to zero when the command is called and returns -1 if error).

unseenMsgNber returns the number of unseen messages in the current mailbox (set to zero when the command is called and returns -1 if error)

 $\textit{mbUID} \ \text{returns the mailbox unique identifier validity value (set to zero when the command is called and returns -1 if error)}.$

With the IMAP4 protocol, the mailbox name is not sufficient to identify a mailbox. As such, a unique identifier validity value is associated with each mailbox. This identifier is particularly valuable for synchronizing tasks.

Thus, you can verify if mailbox "A" has been renamed as "B" or deleted, simply by checking the unique identifier validity value.

On the other hand, this identifier allows you to check whether a mailbox named "A" has been deleted and if another "A" mailbox has been created.

See Also

IMAP_GetFlags, IMAP_ListMBs, IMAP_SetCurrentMB, IMAP_SetFlags.

IMAP_SetCurrentMB (imap_ID; mbName; msgNber; newMsgNber; customFlags; permanentFlags; mbUID) → Integer

Parameter	Type		Description	
imap_ID	Longint	\rightarrow	Reference to an IMAP login	
mbName	Text	\rightarrow	Name of the mailbox to select	
msgNber	Longint	\leftarrow	Number of messages in the current mailbox	
newMsgNber	Longint	\leftarrow	Number of messages with the \Recent flag set	
customFlags	Text	\leftarrow	List of flags currently used for the mailbox	
permanentFlags	s Text	\leftarrow	List of flags permanently modifiable	
mbUID	Longint	\leftarrow	Mailbox unique identifier value	

Function result Integer ← Error code

Description

The IMAP_SetCurrentMB command allows you to open a session (i.e. selects the current working Mailbox) in order to manage the messages of the specified mailbox.

Only one session can be opened at a time during a connection; simultaneous access to multiple mailboxes requires multiple connections (multiple IMAP_Login). The IMAP_SetCurrentMB command automatically closes the current session before attempting the new selection. Consequently, if a mailbox is defined as current and an IMAP_SetCurrentMB command fails, there will no longer be any mailbox defined as current.

You can close a session, (i.e. close the current mailbox) without selecting a new one, by executing the IMAP_SetCurrentMB command using a non-existing mbName and while managing the returned error, either by executing the IMAP_CloseCurrentMB, or by executing the IMAP_Logout command.

imap_ID is a long integer reference to an open connection created with IMAP_Login.

mbName is the full name of an existing mailbox to be defined as current.

msgNber returns the number of messages in the current mailbox (set to zero when IMAP_SetCurrentMB is called and returns -1 if error).

newMsgNber returns the number of recent messages in the current mailbox (set to zero when IMAP_SetCurrentMB is called and returns -1 if error).

customFlags returns the complete list of flags used in the current mailbox. Note that only flags listed in the permanentFlags string can be modified.

permanentFlags returns the list of mailbox message flags that can be changed permanently (except for the \Recent flag, which is managed by the IMAP server). (Set to null string when IMAP_SetCurrentMB is called). Note that the permanentFlags string can also include the special flag *, which means that keywords can be created by trying to store those flags in the mailbox (see IMAP_SetFlags).

If permanentFlags returns a null string, this means that all the flags listed in the customFlags parameter can be permanently changed.

mbUID returns a unique identifier validity value for the current mailbox.

This identifier can be particularly useful if a mailbox is deleted and a new mailbox with the same name is created at a later date. Since the name is the same, a client may not know that this is a new

See Also

IMAP_CloseCurrentMB, IMAP_GetFlags, IMAP_GetMBStatus, IMAP_ListMBs, IMAP_Logout, IMAP_SetFlags.

 $IMAP_GetCurrentMB\ (imap_ID;\ mbName) \longrightarrow Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
mbName	Text	\leftarrow	Name of the current mailbox

Function result Integer \leftarrow Error code

Description

The <code>IMAP_GetCurrentMB</code> command returns the current working mailbox name.

imap_ID is a long integer reference to an open connection created with IMAP_Login.

mbName returns full name of the current mailbox. If the mbName value is a null string, no mailbox is currently selected.

See Also

IMAP_CloseCurrentMB, IMAP_SetCurrentMB.

version 6.8.1

 $IMAP_CloseCurrentMB\ (imap_ID) \longrightarrow Integer$

ParameterTypeDescriptionimap_IDLongint→ Reference to an IMAP login

Function result Integer \leftarrow Error code

Description

The IMAP_CloseCurrentMB command closes the current working mailbox without selecting another mailbox or executing an IMAP_Logout. IMAP_CloseCurrentMB permanently removes all messages that have the \Deleted flag set.

Note: IMAP allows users to work concurrently with the same mailbox in a client/server mode. Suppose that someone carries out synchronization and keeps the connection open, the last mailbox used will remain in selected mode. Anyone else who tries to use this mailbox will not have valid information, or will not be able to work properly, depending on the server implementation, even if the user works in "disconnected mode" (i.e. connected but working using data).

imap_ID is a long integer reference to an open connection created with IMAP_Login.

See Also

IMAP_Delete, IMAP_GetCurrentMB, IMAP_SetCurrentMB, IMAP_SetFlags.

IMAP_Delete

version 6.8.1

 $IMAP_Delete~(imap_ID;~startMsg;~endMsg) \longrightarrow Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
startMsg	Longint	\rightarrow	Start message number
endMsg	Longint	\rightarrow	End message number

Function result Integer \leftarrow Error code

Description

The <u>IMAP_Delete</u> command sets the **\Deleted** flag for the startMsg to endMsg range of messages and then deletes all messages where the **\Deleted** flag is set (including messages where the **\Deleted** flag has previously been set for the current session). Deletion is executed by the IMAP server and takes place when closing the connection (<u>IMAP_Logout</u>) or selecting another current mailbox (<u>IMAP_SetCurrentMB</u>) or closing the current mailbox (<u>IMAP_CloseCurrentMB</u>).

If you do not want to delete right away, you can use the IMAP_SetFlags command and set the \Deleted flag to delete messages later.

imap_ID is a long integer reference to an open connection created with IMAP_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 IMAP_Delete, IMAP_MsqLstInfo, IMAP_MsqLst, IMAP_SetFlags, IMAP_GetFlags and IMAP_CopyToMB commands do not return an error if the startMsg is greater than the endMsg. In the event that this occurs, the command – in effect – does nothing.

See Als

IMAP_CloseCurrentMB, IMAP_Logout, IMAP_SetCurrentMB, IMAP_SetFlags.

IMAP_MsgInfo

version 6.8.1

 $IMAP_MsgInfo\ (imap_ID;\ msgNum;\ msgSize;\ uniqueID) {\longrightarrow}\ Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
msgNum	Longint	\rightarrow	Message number
msgSize	Longint	\leftarrow	Message size
uniqueID	Longint	\leftarrow	Unique ID of message on server

Function result Integer \leftarrow Error code

Description

The <code>IMAP_MsgInfo</code> command returns information about the message identified by <code>msgNum</code> within the currently selected mailbox. Information about the size of the message and its Unique ID will be returned.

imap_ID is a long integer reference to an open connection created with IMAP_Login.

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

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

uniqueID is a long integer variable denoting the Unique ID of the message on the server. The uniqueID is a value assigned to the message by the IMAP4 server software. This value will not change from session to session in the same way as msgNum. The uniqueID value is a good reference to verify if your database has already downloaded a message from the server

See Also

IMAP_Login, IMAP_SetCurrentMB.

IMAP_GetMessage (imap_ID; msgNum; offset; length; msgPart; msgText{; updateSeen}) → Integer

Parameter	Type		Description	
imap_ID	Longint	\rightarrow	Reference to an IMAP login	
msgNum	Longint	\rightarrow	Message number	
offset	Longint	\rightarrow	Offset of character at which to begin retrieval	
length	Longint	\rightarrow	How many characters to return	
msgPart	Integer	\rightarrow	0 = Entire message, 1 = Only header, 2= Only Body	
msgText	Text	\leftarrow	Message Text	
updateSeen	Integer	\rightarrow	0 = Update \Seen Flag; 1 = Do not update	

Function result Integer \leftarrow Error code

Description

The IMAP_GetMessage command returns the complete text of the message identified by msgNum within the current mailbox referenced by IMAP_SetCurrentMB. Unless otherwise specified by the IMAP_SetPrefs command, any linefeed characters within the message will be removed.

The IMAP_GetMessage command returns either the entire block of the message, including header information, or with header only or body only, depending on the msgPart parameter.

imap_ID is a long integer reference to an open connection created with IMAP_Login.

msgNum is a long integer value indicating which message in the mailbox to retrieve. This number represents the position of a message within the current list of messages. You cannot rely on msgNum 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 specified msgPart 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 is limited to 32,000 characters, the *length* parameter should be set to any number below 32,000. Messages whose *msgPart* size is greater than 32K must be saved to the disk via the *IMAP_Download* command.

 ${\it msgPart}$ indicates the message part to retrieve. Values 0, 1 or 2 can be passed:

- 0 = Entire message,
- 1 = Only header,
- 2 = Only body (means first Text/ plain encountered).

Retrieving entire message or only header retrieves raw text without decoding. On the other hand, when retrieving only body, the text will be decoded and converted automatically if needed (see <u>POP3_Charset</u> for more information concerning decoding and conversion rules).

updateSeen is an integer value that indicates if the flag \Seen has to be added to the message flags whether implicitly or not. This parameter is optional and the default value is used if this parameter is not passed.

- 0 = Add \Seen Flag (default value);
- 1= Do not add \Seen Flag;

msgText is a text variable that will receive the retrieved text.

See Also

IMAP_Download, IMAP_Login, IMAP_SetCurrentMB, IMAP_SetPrefs

IMAP_MsgLstInfo (imap_ID; startMsg; endMsg; msgSizeArray; msgNumArray; msgIdArray) → Integer

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
startMsg	Longint	\rightarrow	Start message number
endMsg	Longint	\rightarrow	End message number
msgSizeArray	Longint Array	←	Array of sizes
msgNumArray	Longint Array	←	Array of message numbers
msgIdArray	Longint Array	←	Array of Unique Msg IDs

Function result Integer \leftarrow Error code

Description

The <u>IMAP_MsgLstInfo</u> command returns information about a set of messages in the current working mailbox (defined by the <u>IMAP_SetCurrentMB</u> command). The information is returned into three arrays with each element of the arrays corresponding to one message. Information is returned about the message size and number. The arrays passed as parameters must be of pre-declared types, though they may be of any size. The <u>IMAP_MsgLstInfo</u> command will reset the size of each array to the number of messages retrieved.

The IMAP_MsgLstInfo command will not return an error number if it fails to retrieve information on a message within the current message list. If an error occurs, 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.

imap_ID is a long integer reference to an open connection created with IMAP_Login.

startMsg is a long integer number that 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 current working mailbox.

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 current working mailbox.

Note: The IMAP_Delete, IMAP_MsqLstInfo, IMAP_MsqLst, IMAP_SetFlags, IMAP_GetFlags and IMAP_CopyToMB commands do not return an error if the startMsg is greater than the endMsg. In the event that this occurs, the command – in effect – does nothing.

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

 ${\it msgNumArray} \ {\rm is\ a\ long\ integer\ array\ returned\ containing\ the\ message\ numbers\ between\ startMsg\ and\ endMsg.}$

msgIdArray is a long integer array returning the Unique IDs of the messages between startMsg and endMsg.

See Also

IMAP_SetCurrentMB.

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

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
startMsg	Longint	\rightarrow	Start message number
endMsg	Longint	\rightarrow	End message number
msgHeaderArray	Str Txt Array	\rightarrow	Array of headers to retrieve
msgNumArray	Longint Array	←	Array of message numbers
msgIdArray	Longint Array	\leftarrow	Array of Unique Msg IDs
msgValueArray	2D Str Txt Array	\leftarrow	2D Array of header values

Function result Integer \leftarrow Error code

Description

The <u>IMAP_MsgLst</u> command is used to get specific information of mailbox contents. It allows the user to request specific columns of the message list. This command can only return header item values; it cannot be used to retrieve the body of a message. Header content is automatically decoded and converted if needed (see <u>POP3_Charset</u> for more information concerning the decoding and conversions rules).

 $imap_ID$ is a long integer reference to an open connection created with $\underline{IMAP_Login}$.

startMsg is a long integer number that 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 current working mailbox.

endMsg is a long integer number that 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 current working mailbox.

Note: The <u>IMAP_Delete</u>, <u>IMAP_MsgLstInfo</u>, <u>IMAP_MsgLst</u>, <u>IMAP_MsgLst</u>, <u>IMAP_SetFlags</u>, <u>IMAP_GetFlags</u> and <u>IMAP_CopyToMB</u> commands do not return an error if the *startMsg* is greater than the *endMsg*. In the event that this occurs, the command – in effect – does nothing.

msgHeaderArray is a string or text array that lists the specific e-mail headers you wish to retrieve.

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

msgIdArray is a long integer array returning the Unique IDs of the messages between startMsg and endMsg.

msgValueArray is a 2-dimensional array that receives the data for each header specified in msgHeaderArray. Each requested header will have a matching array in the first dimension of valueArray.

Example

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

aValues{1}{1} may equal "Thu, 19 November 1998 00:24:02 -0800"

aValues{2}{1} may equal "Jack@4d.com"

aValues{3}{1} may equal "Call your wife"

Errors are handled the following manner:

1) Only communication-related error codes will be returned. If the command cannot 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 a message does not constitute an error:
- An array element for the message will be created.
- $\bullet \ The \ msgNumArray \ and \ msgIDArray \ array \ elements \ will \ contain \ the \ appropriate \ values.$



version 6.8.1

IMAP_SetFlags (imap_ID; startMsg; endMsg; msgFlagsList; deleteOption)

Integer

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
startMsg	Longint	\rightarrow	Start message number
endMsg	Longint	\rightarrow	End message number
msgFlagsList	Str Txt	\rightarrow	Flag values to add or remove
deleteOption	Integer	\rightarrow	1 = add flag value, $0 = $ remove flag value

Function result Integer - Error code

Description

The IMAP_SetFlags command allows adding or clearing of several flags at a time, attached to the specified range of messages.

IMAP protocol allows associating of a list of flags with a message. There are two types of flags: **permanent** or **session-only**.

Permanent flags are added or removed permanently from the message flags (see IMAP_SetCurrentMB); in other words, subsequent sessions will reflect any changes in permanent flags.

Changes made for session flags are only valid for that session.

The currently defined system flags are:

- Seen: Message has been read.
- · Answered: Message has been answered.
- Flagged: Message is "flagged" for urgent/special attention.
- Deleted: Message is "deleted" for later removal with IMAP_Delete, IMAP_CloseCurrentMB, IMAP_SetCurrentMB or IMAP_Logout.
- Draft: Message is in draft format; in other words, not complete.
- Recent: Message "recently" arrived in this mailbox. This session is the first session notified about this message; subsequent sessions will not see the \Recent flag set for this message. This permanent flag is managed by the IMAP server and cannot be modified by an IMAP client using IMAP_SetFlags, for instance.

An IMAP server may allow a client to define new "flags" or, on another IMAP server, may allow managing of flags other than those previously indicated. This depends on the IMAP server implementation. In this case, these special flags are called "keywords" and do not begin with "\" (see <u>IMAP SetCurrentMB</u>).

Note: If you set the \Deleted flag and close the current session by executing IMAP_SetCurrentMB, IMAP_Logout, the message will be "deleted" permanently. Image: imap_ID is a long integer reference to an open connection created with IMAP_Login.

startMsg is a long integer number that 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 current working mailbox.

endMsg is a long integer number that 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 current working mailbox.

Note: The IMAP Delete, IMAP MsqLstInfo, IMAP MsqLst, IMAP_SetFlags, IMAP GetFlags and IMAP CopyToMB commands do not return an error if the startMsg is greater than the endMsg. In the event that this occurs, the command – in effect – does nothing.

msgFlagsList may contain one or several flags. In the case of several flags, the string must be a list of flags, separated by spaces. See examples below.

Only flags listed as permanentFlags, (see $\underline{IMAP_SetCurrentMB}$), will be applied.

deleteOption is an integer value which specifies whether to remove or add the flag(s) specified by the msgFlagsList parameter:

- $\bullet \ A \ value \ of \ zero \ will \ remove \ the \ flag(s) \ specified \ in \ \textit{msgFlagsList}.$
- \bullet A value of 1 will add the flag(s) specified in msgFlagsList.

Examples

 $1. \ \, \text{Set the $$^{\text{Nnswered}}$ and $$^{\text{Draft}}$ flags for the messages specified by $$startMsg$ and $endMsg$ whether these flags were set previously or not:} \\$

```
msgFlagsName:="\Answered \Draft"
  `\Answered and \Draft are separated by a space (ASCII code)
IMAP_SetFlags (imap_ID;startMsg;endMsg;msgFlagsName;1)
```

2. Remove the **Deleted** flag for the messages specified by *startMsg* and *endMsg* whether this flag was set previously or not:

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

3. Set the \Deleted flag for the messages specified by startMsg and endMsg whether this flag was set previously or not:

```
msgFlagsName:="\Deleted"
IMAP_SetFlags (imap_ID; startMsg; endMsg; msgFlagsName; 1)
```

```
4. Set the \Answered flag depending on the \CheckBoxAnswered value:
  $Error:= IMAP_SetFlags (vImap_ID;$msgNum;$msgNum;"\Answered";Num (CheckBoxAnswered =0))
IMAP GetFlags, IMAP SetCurrentMB.
```

 $IMAP_GetFlags\ (imap_ID;\ startMsg;\ endMsg;\ msgFlagsArray;\ msgNumArray) \longrightarrow Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
startMsg	Text	\rightarrow	Start message number
endMsg	Text	\rightarrow	End message number
msgFlagsArray	Str Txt Array	←	Flag values for each message
msgNumArray	Longint Array	←	Array of message numbers

Function result Integer ← Error code

Description

The $\ensuremath{\textit{IMAP_GetFlags}}$ command returns the list of flags for the specified messages.

imap_ID is a long integer reference to an open connection created with IMAP_Login.

startMsg is a long integer number that 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 current working mailbox.

endMsg is a long integer number that 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 current working mailbox.

Note: The <u>IMAP_Delete</u>, <u>IMAP_MsqLstInfo</u>, <u>IMAP_MsqLst</u>, <u>IMAP_SetFlags</u>, <u>IMAP_GetFlags</u> and <u>IMAP_CopyToMB</u> commands do not return an error if the startMsg is greater than the endMsg. In the event that this occurs, the command — in effect — does nothing.

msgFlagsArray is a string or text array returned containing the list of flags, separated by spaces, of each message number between startMsg and endMsg.

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

See Also

IMAP_SetCurrentMB, IMAP_SetFlags.

IMAP_MsgFetch (imap_ID; msgNum; msgDataItem; msgDataItemValue) -> Integer

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
msgNum	Longint	\rightarrow	Message number
msgDataItem	Text	\rightarrow	Data item(s) to retrieve
msgDataItemValue	e Text	\leftarrow	Data item(s) value

Function result Integer ← Error code

Description

The IMAP_MsgFetch command allows the user to request one or several basic data items for the specified message without downloading the message.

imap_ID is a long integer reference to an open connection created with IMAP_Login.

msgNum is a long integer value indicating the message to be checked. This value represents the position of a message within the current list of messages. You cannot rely on the msgNum remaining the same for a specific e-mail item from session to session.

msgDataItem is a text variable that indicates one or several data items you want to retrieve. In the case of several data items, a space character must separate each of them. There are two kinds of data items:

- · Basic data items, which only retrieve one piece of information, and
- Macro data items, which retrieve several pieces of information, issued from basic data items at one time. Three macros specify commonly-used sets of data items and can be used in place of them. A macro must be used by itself, and not in conjunction with other macros or data items.

For more information on data items, see the "Basic data items" and "Macro data items" paragraphs below.

msqDataItemValue is a text variable that can return either a single DataItem/DataItemValue pair or a list of DataItem/DataItemValue pairs depending on the msqDataItem parameter value.

- In the case of a single <code>DataItem/DataItemValue</code>, the returned text structure is as follows: <code>DataItem name+Space+DataItemValue</code>
- In the case of a list of DataItem/DataItem/alue pairs, the returned text structure is as follows: DataItem name1+Space+DataItemValue1+Space+DataItem name2+Space+DataItemValue1.

 $msgData Item Value \ may \ contain \ a \ parenthesized \ list, \ a \ quoted \ string \ or \ a \ single \ string \ depending \ on \ the \ msgData Item \ parameter.$

 $\bullet \ \, \text{The parenthesized list is structured as follows (see \textit{FLAGS} \ case for example): (\textit{FirstDataItemValue} + space + 2ndDataItemValue) \\$

If the parenthesized list returns only parentheses, this means that there is no item value. This rule does not apply to address parenthesized lists (see ENVELOPE).

 $\bullet \ Quoted \ strings \ are \ structured \ as \ follows \ (see \ \ INTERNAL DATE \ case \ for \ example): \ Data \ Item \ name + Space + Quote + Data \ Item \ Value + Quote + Quote$

If the DataItem value returns "", this means that it is a null string.

• Strings that are not quoted indicate integer, long integer or numeric values and are structured as follows: DataItem name + Space + DataItem Value.

In this case, you will most likely have to convert to the appropriate type (see UID case for example).

Note: Quotes are generally used when the string value includes special characters, such as a space or parentheses. As such, when you parse the resulting string of the IMAP_Fetch command, quote characters are taken into consideration when processing the string content.

Basic data items

• INTERNALDATE

Retrieves the internal date and time of the message on the IMAP server. This is not the date and time returned by the "Date" header, but rather a date and time that indicate when the message was received. For messages delivered via an SMTP server, this date usually reflects the date and time of the final delivery of the message. For messages sent after an *IMAP_Copy* command, this data usually reflects the internal date and time of the source message.

INTERNALDATE data item value returns a quoted string.

Example:

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

msgDataItem returns INTERNALDATE "17-Jul-2001 15:45:37 +0200"

• FLAGS

Retrieves the parenthesized list of flags that are set for the specified message. Flags values are separated by spaces.

Example:

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

 $msgDataItem\ returns\ FLAGS\ ()$ if there is no flag set for the specified message.

msgDataItem returns FLAGS (\Seen \Answered) if \Seen and \Answered flags are set for the specified message.

• RFC822.SIZE

Retrieves the number of bytes in the message, as expressed in RFC-822 format. The Data item is separated from the returned value by a space. An unquoted string is returned, which means you will probably need to convert this string into a longint value (see *UID* example).

Example

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

msgDataItem returns RFC822.SIZE 99599

ENVELOPE

Retrieves the parenthesized list describing the header part for the specified message. The server computes this by parsing the message header and defaulting various fields where necessary.

The header fields are returned in the following order: date, subject, from, sender, reply-to, to, cc, bcc, in-reply-to, and message-id. The date, subject, in-reply-to and message-id fields are quoted strings:

ENVELOPE ("date" "subject" (from) (sender) (reply-to) (to) (cc) (bcc) "in-reply-to" "message-id")

Example:

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

msgDataItem returns ENVELOPE ("Tue, 17 Jul 2001 17:26:34 +0200" "Test" (("RSmith" NIL "RSmith" "test")) (("RSmith" NIL "RSmith" NIL "RSmith" NIL "RSmith" "test")) (("RSmith" NIL "RSmith" NIL

```
"Tue, 17 Jul 2001 17:26:34 +0200"
                                                  date header
Date:
              "Test"
Subject:
                                                  subject header
              (("RSmith" NIL "RSmith" "test"))
From:
                                                  address structures
              (("RSmith" NIL "RSmith" "test"))
Sender:
                                                  address structures
              (("RSmith" NIL "RSmith" "test"))
reply-to:
                                                  address structures
              (("RSmith" NIL "RSmith" "test"))
                                                  address structures
to:
                                                  Cc header not used
CC:
                                                  Bcc header not used
bcc:
                                                  In-reply-to header
in-reply-to:
              <ee6b33a.-1@Mail.x6foadRIbnm>" message-id header
message-id:
```

The from, sender, reply-to, to, cc and bcc fields are parenthesized lists of address structures. An address structure is a parenthesized list that describes an electronic mail address. The fields of an address structure are in the following order: personal name, [SMTP] at-domain-list (source route), mailbox name and host name. For instance, (("RSmith" NIL "RSmith" "test")).

(From [RFC-822]) Group syntax is indicated by a special form of address structure where the host name field is NIL. If the mailbox name field is also NIL, this is an end-of-group marker (semi-colon in RFC 822 syntax). If the mailbox name field is non-NIL, this is the start-of-group marker and the mailbox name field holds the group name phrase.

A field of an envelope or address structure that is not applicable is presented as NIL. Note that the server MUST default the reply-to and sender fields from the "from" field; the client is not expected to know how to do this.

• BODY

 $Returns \ the \ same \ information \ as \ BODYSTRUCTURE \ except \ for \ the \ Extension \ data \ (see \ BODYSTRUCTURE) \ which \ is \ not \ returned.$

Example:

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

msgDataItem returns BODY ("TEXT" "PLAIN" ("CHARSET" "us-ascii") NIL NIL "8BIT" 8 1)

· BODYSTRUCTURE

Retrieves the MIME body structure of the message. The server computes this by parsing the MIME header fields in the message header and MIME headers in the body part. This data item is especially useful for scanning a message content without downloading it. For instance, you can quickly check the size of each part or just check the attachment file names. BODYSTRUCTURE returns a parenthesized list including the parenthesized list, quoted strings and unquoted strings.

Depending on the message content, BODYSTRUCTURE will return either a "non- multipart" parenthesized list or a nested one ("multipart" parenthesized list):

• "non-multipart" parenthesized list: this is, for instance, similar to non-multipart e-mail; a simple text message of 48 lines and 2279 bytes can have a body structure of: ("TEXT" "PLAIN" ("CHARSET" "us-ascii") NIL NIL "18 NIL NIL NIL NIL).

The basic fields of a "non-multipart" parenthesized list are in the following order:

body type

A string giving the content media type name

(Content-type: media type e.g. TEXT)

body subtype A string giving the content subtype name

(Content-type: subtype e.g. PLAIN)

body parameter

A parenthesized list of attribute/value pairs

parenthesized list

[e.g. ("CHARSET" "US-ASCII" "NAME" "cc.diff")

where "US-ASCII" is the value of "CHARSET" and

"cc.diff" is the value of "NAME".

body id

A string giving the content id (allows one body to make a

reference

to another one). Accordingly, bodies may be labeled using

the

"Content-ID" header field. The Content-ID value has

special

semantics in the case of a multipart/alternative media type. This is explained in the section of RFC 2046 dealing with

multipart/alternative cases.

body description

A string giving the content description

body encoding

A string giving the content transfer encoding

(Content-Transfer-Encoding)

body size

A number giving the size of the body in bytes.

Note that this is the size during transfer encoding

and not the resulting size after decoding.

Extension data follows the basic fields and type-specific fields listed above. Extension data is never returned with the BODY fetch, but can be returned with a BODYSTRUCTURE fetch

 $Extension \ data\ , if\ present,\ of\ a\ "non\ multipart"\ parenthesized\ list\ MUST\ be\ in\ the\ defined\ order to the present of the pre$

body MD5 A string giving the body MD5 value as defined in [MD5] body disposition A parenthesized list consisting of a disposition type string

followed by a parenthesized list of disposition

attribute/value pairs

as defined in [DISPOSITION]

body language

A string or parenthesized list giving the body language

value

as defined in [LANGUAGE-TAGS]

Any extension data which follows are not yet defined in this version of the protocol and will be as described above under multipart extension data

⁻ A body of the MESSAGE type and RFC822 subtype contains, immediately following the basic fields, the envelope structure, body structure, and size in text lines of the encapsulated message.

⁻ A body of the TEXT type contains, immediately following the basic fields, the size of the body in text lines. Note that this is the size during content transfer encoding and not the resulting size after decoding.

Example:

("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 2279 48 NIL NIL NIL)

Description:

("bodytype" "bodysubtype" (BodyParameterParenthesizedList) bodyId bodyDescription "bodyEncoding" BodySize BodySizeInTextLines ExtensionDataBODYmd5 ExtensionDataBodyDisposition ExtensionDataBodyLanguage)

• "multipart" parenthesized list: this is the case of multipart e-mail; it includes a "non-multipart" parenthesized list.

Parenthesis nesting indicates multiple parts. The first element of the parenthesized list is a nested body instead of a body type. The second element of the parenthesized list is the multipart subtype (mixed, digest, parallel, alternative, etc.).

The multipart subtype is followed by the Extension data. Extension data, if present, MUST be in the defined order:

body parameter A parenthesized list of attribute/value pairs

parenthesized list body disposition

A parenthesized list consisting of a disposition type string

followed by a parenthesized list of disposition

attribute/value pairs

as defined in [DISPOSITION]

body language

A string or parenthesized list giving the body language

value

as defined in [LANGUAGE-TAGS]

Any extension data which follows are not yet defined in this version of the protocol. This extension data can consist of zero or more NILs, strings, numbers or potentially nested parenthesized lists of such data. Client implementations that do a BODYSTRUCTURE fetch MUST be prepared to accept such extension data. Server implementations MUST NOT send such extension data until it has been defined by a revision of this protocol.

Example

Description

(("bodytype" "bodysubtype" (BodyParameterParenthesizedList) bodyId bodyDescription "bodyEncoding" BodySize BodySizeInTextLines ExtensionDataBODYmd5 ExtensionDataBodyDisposition ExtensionDataBodyLanguage) ("bodytype" "bodysubtype" (BodyParameterParenthesizedList) bodyId bodyDescription "bodyEncoding" BodySize BodySizeInTextLines ExtensionDataBODYmd5 ExtensionDataBodyDisposition ExtensionDataBodyLanguage) "multipartSubtype" (ExtensionDataBodyParameterList) ExtensionDataBodyDisposition ExtensionDataBodyLanguage))

· UII

 $Retrieves \ a \ number \ expressing \ the \ message \ unique \ identifier \ of \ the \ message. \ This \ is \ equivalent \ to \ executing \ the \ \underline{\it IMAP_UIDToMsgNum}.$

Since this number is returned into a text area, you will have to convert it into a Long integer.

Example:

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

msgDataItemValue returns UID 250000186

To retrieve a longint value:

```
C LONGINT(vLongint)
VLongint:=Num("250000186")
```

Macro data items

• FAST

Macro equivalent to: (FLAGS INTERNALDATE RFC822.SIZE)

Example:

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

msgDataItemValue returns "FLAGS (\Seen \Answered) INTERNALDATE "17-Jul-2001 15:45:37 +0200" RFC822.SIZE 99599"

· ALL

Macro equivalent to: (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE)

• FUL

Macro equivalent to: (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE BODY)

See Also



 $IMAP_Download\ (imap_ID;\ msgNum;\ headerOnly;\ fileName\{;\ updateSeen\}) \longrightarrow Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
msgNum	Longint	\rightarrow	Message number
headerOnly	Integer	\rightarrow	0 = Entire message, $1 = $ Only header
fileName	Text	\rightarrow	Local Filename
		\leftarrow	Resulting Local Filename
updateSeen	Integer	\rightarrow	0 = Add \Seen Flag; 1= Do not add \Seen Flag

Function result Integer \leftarrow Error code

Description

The <code>IMAP_Download</code> command is designed to retrieve a message from an IMAP server by downloading it to a disk-based file. Any IMAP message which contains attachments or whose size is greater than 32K should be downloaded with this command. File attachments can only be extracted from the messages retrieved in this way.

 $imap_ID$ is a long integer reference to an open connection created with $\underline{IMAP_Login}$.

msgNum is a long integer value indicating which message in the mailbox to retrieve. msgNum represents the position of a message within the current list of messages. You cannot rely on the msgNum 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 in three different ways:

- "" = Saves the file in the folder set by <u>IMAP_SetPrefs</u>, 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 $\underline{\mathit{IMAP_SetPrefs}}$ entitled 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 <u>IMAP_SetPrefs</u>, the message will be saved in the same folder as the structure of the database (with 4D single-user) or in the 4D Client folder (with 4D Server).

After the file has been saved to the disk, the final name of the file will be returned to the variable passed as the fileName parameter. If you attempt to call IMAP_Download with a fileName that already exists within the download folder, the name will be numerically incremented and its new value as saved to the disk will be returned to the fileName variable.

updateSeen is an integer value that indicates if the \Seen flag must be added to the message flags list, whether implicitly or not. This parameter is optional and a default value is used if this parameter is not passed:

- $0 = Add \setminus Seen Flag$
- 1 = Do not add \Seen Flag

Default value is set to 0 which implicitly means add $\ensuremath{\backslash} \mathbf{Seen}$ flag.

See Also

IMAP_GetMessage, IMAP_SetPrefs.

IMAP_UIDToMsgNum

version 6.8.1

 $IMAP_UIDToMsgNum\ (imap_ID; unique_ID; msgNum) \longrightarrow Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
unique_ID	Longint	\rightarrow	Message unique identifier value
msgNum	Longint	\leftarrow	Message number

Function result Integer \leftarrow Error code

Description

The <code>IMAP_UIDToMsgNum</code> command converts a message <code>unique_ID</code> value to its current <code>msgNum</code> within the list of messages in the current mailbox referenced by <code>imap_ID</code>. Since a specific e-mail message <code>msgNum</code> is a floating value relative to other items in the electronic mail list, this command returns the current position of a message whose information may have been retrieved during a prior IMAP session.

imap_ID is a long integer reference to an open connection created with IMAP_Login.

unique_ID is a long integer value indicating the **Unique ID** of a message to be located on the IMAP server.

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

See Als

 $\underline{IMAP_MsgNumToUID}, \underline{IMAP_SetCurrentMB}.$

IMAP_MsgNumToUID

version 6.8.1

 $IMAP_MsgNumToUID\ (imap_ID;\ msgNum;\ unique_ID) {\:\longrightarrow\:} Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
msgNum	Longint	\rightarrow	Message number
unique_ID	Longint	\leftarrow	Message unique identifier value

Function result Integer \leftarrow Error code

Description

The IMAP_MsgNumToUID command converts a message number within the list of messages in the current mailbox referenced by imap_ID to its current unique_ID value.

imap_ID is a long integer reference to an open connection created with <u>IMAP_Login</u>.

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

unique_ID is a long integer value returning the Unique ID of a message to be located on the IMAP server.

See Als

 $\underline{IMAP_SetCurrentMB}, \underline{IMAP_UIDToMsgNum}.$

version 6.8.1

IMAP Search (imap ID; searchCriteria; msgNumArray) -> Integer

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
searchCriteria	Text	\rightarrow	Search criteria
msgNumArray	Longint Array	\leftarrow	Array of message numbers

Function result Integer \leftarrow Error code

Description

The IMAP_Search command searches for messages that match the given search criteria in the current mailbox. searchCriteria consists of one or more search keys. msgNumArray returns a listing of message sequence numbers corresponding to those messages that match the search criteria.

msgNumArray returns a listing of message sequence numbers corresponding to those messages that match the searching criteria.

imap_ID is a long integer reference to an open connection created with IMAP_Login

search Criteria is a text parameter listing one or more search keys (see "Authorized search keys" at the end of this paragraph) associated or not with values to look for. A search key may be a single item or can also be a parenthesized list of one or more search keys. For example:

SearchKey1 = FLAGGED

SearchKev2 = NOT FLAGGED

 $SearchKey3 = FLAGGED\ DRAFT$

Note: Matching is usually not case-sensitive.

• If the searchCriteria is a null string, the search will be equivalent to a "select all":

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

- ... returns all messages in the current working mailbox.
- If the searchCriteria includes multiple search keys, the result is the intersection (AND function) of all the messages that match those keys.

searchCriteria = FLAGGED FROM "SMITH"

- ... returns all messages with \Flagged flag set AND sent by Smith.
- You can use the **OR** or **NOT** operators as follows:

searchCriteria = OR SEEN FLAGGED

... returns all messages with $\ensuremath{\backslash} \mathbf{Seen}$ flag set OR $\ensuremath{\backslash} \mathbf{Flagged}$ flag set

searchCriteria = NOT SEEN

... returns all messages with $\ensuremath{\backslash} \mathbf{Seen}$ flag not set .

searchCriteria = HEADER CONTENT-TYPE "MIXED" NOT HEADER CONTENT-TYPE "TEXT"...

... returns message whose content-type header contains "Mixed" and does not contain "Text".

searchCriteria = HEADER CONTENT-TYPE "E" NOT SUBJECT "o" NOT HEADER CONTENT-TYPE "MIXED"

 $...\ returns\ message\ whose\ content-type\ header\ contains\ "e"\ and\ whose\ Subject\ header\ does\ not\ contain\ "o"\ and\ whose\ content-type\ header\ is\ not\ "Mixed".$

 $search Criteria = OR \ (ANSWERED \ SMALLER \ 400) \ (HEADER \ CONTENT-TYPE \ "E" \ NOT \ SUBJECT \ "o" \ NOT \ HEADER \ CONTENT-TYPE \ "MIXED")$

 \ldots returns messages matching the first parenthesized list OR the second one.

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

... returns messages matching with \Answered flag set OR messages whose size is smaller than 400 bytes AND the parenthesized list as specified.

As concerns the last two examples, notice that the result of the search is different when you remove the parentheses of the first search key list.

• The searchCriteria may include the optional [CHARSET] specification. This consists of the "CHARSET" word followed by a registered [CHARSET] (US ASCII, ISO-8859). It indicates the charset of the searchCriteria string. Therefore, you must convert the searchCriteria string into the specified charset if you use the [CHARSET] specification (see the 4th Dimension Mac to ISO command).

By default, 4D Internet Commands encode in Quotable Printable the searchCriteria string if it contains extended characters.

searchCriteria = CHARSET "ISO-8859" BODY "Help"

... means the search criteria uses the charset iso-8859 and the server will have to convert the search criteria before searching, if necessary.

Search value types

Search-keys may request the value to search for:

Search-keys with a date value

<date> is a string that must be formatted as follows: date-day+"-"+date-month+"-"+date-year where date-day indicates the number of the day of the month (max. 2 characters), date-month indicates the name of the month (Jan/Feb/Mar/Apr/May/Jun/Jul/Aug/Sep/Oct/Dec) and date-year indicates the year (4 characters).

Example: searchCriteria = SENTBEFORE 1-Feb-2000 (a date does not usually need to be quoted since it does not contain any special characters)

· Search-keys with a string value

<string> may contain any character and must be quoted. If the string does not contain any special characters, like the space character for instance, it does not need to be quoted. Quoting such strings will ensure that your string value will be correctly interpreted.

Example: searchCriteria = FROM "SMITH"

Note: For all search keys that use strings, a message matches the key if the string is a substring of the field. Matching is not case-sensitive.

· Search-keys with a field-name value

<field-name> is the name of a header field.

Example: searchCriteria = HEADER CONTENT-TYPE "MIXED"

· Search-keys with a flag value

<flag> may accept one or several keywords (including standard flags), separated by spaces.

Example: searchCriteria = KEYWORD \Flagged \Draft

· Search-keys with a message set value

Identifies a set of messages. For message sequence numbers, these are consecutive numbers from 1 to the total number of messages in the mailbox.

A comma delimits individual numbers; a colon delimits between two numbers inclusive.

Examples:

2,4:7,9,12:* is 2,4,5,6,7,9,12,13,14,15 for a mailbox with 15 messages.

searchCriteria = 1:5 ANSWERED search in message selection from message sequence number 1 to 5 for messages which have the \Answered flag set.

searchCriteria= 2,4 ANSWERED search in the message selection (message numbers 2 and 4) for messages which have the \Answered flag set.

Authorized search-keys

ALL: All messages in the mailbox.

ANSWERED: Messages with the \Answered flag set.

UNANSWERED: Messages that do not have the \Answered flag set.

DELETED: Messages with the **Deleted** flag set.

 $\boldsymbol{UNDELETED} : Messages that do not have the \mathbb{Deleted}$ flag set.

DRAFT: Messages with the \Draft flag set.

UNDRAFT: Messages that do not have the \Draft flag set.

 $\label{eq:FLAGGED:Messages} \textbf{FLAGGED: Messages with the $\ensuremath{\backslash} \textbf{Flagged}$ flag set.}$

UNFLAGGED: Messages that do not have the \Flagged flag set.

RECENT: Messages that have the \Recent flag set.

OLD: Messages that do not have the \Recent flag set.

SEEN: Messages that have the \Seen flag set.

 $\mathbf{UNSEEN} : \mathbf{Messages}$ that do not have the **\Seen** flag set.

NEW: Messages that have the **\Recent** flag set but not the **\Seen** flag. This is functionally equivalent to "(RECENT UNSEEN)".

KEYWORD <flag>: Messages with the specified keyword set.

 $\label{lem:unkeyword} \textbf{UNKEYWORD} < \textbf{flag} >: \textbf{Messages that do not have the specified keyword set.}$

 $\label{eq:BEFORE} \textbf{Seeds: Messages whose internal date is earlier than the specified date.}$

ON <date>: Messages whose internal date is within the specified date.

 $\textbf{SINCE} < \!\! \textbf{date} \!\! >: \textbf{Messages whose internal date is within or later than the specified date}.$

SENTBEFORE < date >: Messages whose Date header is earlier than the specified date.

SENTON <date>: Messages whose Date header is within the specified date.

SENTSINCE < date >: Messages whose Date header is within or later than the specified date.

TO <string>: Messages that contain the specified string in the TO header.

FROM <string>: Messages that contain the specified string in the FROM header.

 $\label{eq:cc_string} \textbf{CC} < \!\! \textbf{string} \!\! > \!\! : \text{Messages that contain the specified string in the CC header.}$

BCC <string>: Messages that contain the specified string in the BCC header.

 ${\bf SUBJECT\,{<}string{>}}: Messages \ that \ contain \ the \ specified \ string \ in \ the \ Subject \ header.$

BODY <string>: Messages that contain the specified string in the message body.

TEXT <string>: Messages that contain the specified string in the header or in the message body.

HEADER <field-name > <string>: Messages that have a header with the specified field-name and that contain the specified string in the field-body.

UID <message UID>: Messages with unique identifiers corresponding to the specified unique identifier set.

LARGER < n >: Messages with a size larger than the specified number of bytes. $\label{eq:smaller} \textbf{SMALLER} < \textbf{n} >: \text{ Messages with a size smaller than the specified number of bytes.}$ **NOT** <**search-key**>: Messages that do not match the specified search key. OR <search-key1> <search-key2>: Messages that match either search key. See Also IMAP_GetFlags, IMAP_SetFlags.

 $IMAP_CopyToMB\ (imap_ID;\ startMsg;\ endMsg;\ mbNameTarget\{;\ msgDelete\}) \longrightarrow Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
startMsg	Longint	\rightarrow	Start message number
endMsg	Longint	\rightarrow	End message number
mbNameTarge	et Text	\rightarrow	Name of the destination mailbox
msgDelete	Integer	\rightarrow	0= Do not remove from source mailbox,
			1= Remove from source mailbox

Function result Integer \leftarrow Error code

Description

Given a range of messages from startMsg to endMsg, the IMAP_CopyToMB command will copy the specified message(s) to the end of the specified mbNameTarget destination mailbox. The flags and the internal date of the message(s) are usually preserved in the destination mailbox, depending on the IMAP server implementation.

After being copied, original messages are not removed from the source mailbox. If you want to remove them, you can use one of the 3 following processes:

- use the IMAP_Delete command,
- set the msgDelete optional parameter to 1,
- \bullet set $\underline{\mathit{IMAP_SetFlags}}$ (\Deleted): the messages will be removed when the session is closed.

Note: The msgDelete parameter will force the execution of an IMAP_Delete; therefore, the deletion will include messages between startMsg and endMsg and ALL messages for which the \Deleted flag is set.

If the destination mailbox does not exist, an error is returned.

imap_ID is a long integer reference to an open connection created with IMAP_Login.

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

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

Note: The IMAP_Delete, IMAP_MsqLstInfo, IMAP_MsqLst, IMAP_SetFlags, IMAP_GetFlags and IMAP_CopyToMB commands do not return an error if the startMsg is greater than the endMsg. In the event that this occurs, the command – in effect – does nothing.

mbNameTarget is the full name of the mailbox where the specified messages will be copied.

 $The \textit{ msgDelete} \ optional \ parameter \ allows \ setting \ if \ you \ want \ to \ remove \ the \ message \ from \ the \ source \ mailbox.$

- ullet 0= Do not remove from source mailbox (default value);
- 1= Remove from source mailbox.

If msgDelete is omitted, the default value is used.

If copying fails, the message is not removed from the source mailbox.

If the user does not have access to remove messages, an error message is generated.

See Also

IMAP_CreateMB, IMAP_ListMBs, IMAP_RenameMB.

version 6.8.1

 $IMAP_CreateMB\ (imap_ID;\ mbName) {\longrightarrow}\ Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
mbName	Text	\rightarrow	Name of the mailbox to be created

Function result Integer \leftarrow Error code

Description

The IMAP_CreateMB command creates a mailbox with the given name. If the IMAP server's hierarchy separator character appears elsewhere in the mailbox name, the IMAP server will create any parent names needed to create the given mailbox.

In other words, an attempt to create "Projects/IMAP/Doc" on a server in which "/" is the hierarchy separator character will create:

- Only the "Doc" mailbox if "Projects" & "IMAP" already exist.
- "IMAP" & "Doc" mailboxes if only "Projects" already exists.
- "Projects" & "IMAP" & "Doc" mailboxes, if they do not already exist.

imap_ID is a long integer reference to an open connection created with <u>IMAP_Login</u>.

mbName is the full name of the mailbox to be created (see naming rules in the IMAP introduction).

Note: Attempting to create an INBOX (which is a special name reserved to mean "the primary mailbox for this user on this server") or a mailbox with a name referring to an existing mailbox will lead to an error.

See Also

IMAP_ListMBs, IMAP_RenameMB.

 $IMAP_DeleteMB\ (imap_ID;\ mbName) {\longrightarrow}\ Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
mbName	Text	\rightarrow	Mailbox name to be deleted

Function result Integer \leftarrow Error code

Description

The IMAP_DeleteMB command permanently removes the mailbox of a given name. Attempting to delete an INBOX or a mailbox that does not exist will generate an error.

The IMAP_DeleteMB command cannot delete a mailbox which has child mailboxes and which also has the \Noselect mailbox attribute.

It is possible to delete a mailbox that has child names and does not have the \Noselect mailbox attribute. In this case, all messages in the mailbox are removed and it acquires the \Noselect mailbox attribute

Note: The IMAP protocol does not guarantee that you can delete a mailbox which is not empty, though on some servers this is allowed. If you do choose to attempt it, you must be prepared to use another method should the more convenient one fail. Further, you should not try to delete the current working mailbox while it is open, but should first close it; some servers do not permit deletion of the current working mailbox while it is open, but should first close it; some servers do not permit deletion of the current working mailbox while it is open, but should first close it; some servers do not permit deletion of the current working mailbox while it is open, but should first close it; some servers do not permit deletion of the current working mailbox while it is open, but should first close it; some servers do not permit deletion of the current working mailbox while it is open, but should first close it; some servers do not permit deletion of the current working mailbox while it is open, but should first close it; some servers do not permit deletion of the current working mailbox while it is open, but should first close it; some servers do not permit deletion of the current working mailbox while it is open, but should first close it; some servers do not permit deletion of the current working mailbox while it is open, but should first close it; some servers do not permit deletion of the current working mailbox while it is open, but should first close it; some servers do not permit deletion of the current working mailbox while it is open, but should first close it; some servers do not permit deletion of the current working mailbox while it is open, but should first close it; some servers do not permit deletion of the current working mailbox while it is open.

imap_ID is a long integer reference to an open connection created with <u>IMAP_Login</u>.

mbName is the full name of the mailbox to be deleted.

See Also

 $\underline{IMAP_CloseCurrentMB}, \underline{IMAP_Delete}, \underline{IMAP_SetCurrentMB}.$

IMAP_RenameMB

version 6.8.1

 $IMAP_RenameMB \ (imap_ID; \ mbName; \ newMBName) \longrightarrow Integer$

Parameter	Type		Description
imap_ID	Longint	\rightarrow	Reference to an IMAP login
mbName	Text	\rightarrow	Name of the mailbox to be renamed
newMBName	Text	\rightarrow	New mailbox name

Function result Integer ← Error code

Description

The IMAP_RenameMB command changes the name of a mailbox. Attempting to rename a mailbox from a mailbox name that does not exist or to a mailbox name that already exists will generate an error.

Note: Renaming an INBOX is permitted and involves special behavior. It moves all messages in the INBOX to a new mailbox with the given name, leaving the INBOX empty. If the server implementation allows child names for the INBOX, these are unaffected by its renaming.

 $imap_ID$ is a long integer reference to an open connection created with $\underline{IMAP_Login}$.

mbName is the full name of the mailbox to be renamed (see naming rules in the IMAP introduction).

newMBName is the full name to apply to rename the mbName mailbox.

See Also

IMAP_CreateMB, IMAP_ListMBs, IMAP_Login.

 $IMAP_Logout\ (imap_ID) \longrightarrow Integer$

Parameter	Type	
imap_ID	Longint	

Description

- → Reference to an IMAP login
- \leftarrow 0 = Command successfully logs off

Function result Integer \leftarrow Error code

Description

The <u>IMAP_Logout</u> command will log off of the open IMAP connection referred to by the <u>imap_ID</u> variable. If the command successfully logs off the IMAP server, a zero value is returned back as the current <u>imap_ID</u>.

Note: Closing a connection automatically closes the current session.

imap_ID is a long integer reference to an open connection created with IMAP_Login.

See Also

IMAP_Login.

IC Downloaded Mail

- Downloading Mail, Overview

 MSG SetPrefs (stripLineFeed; msgFolder; attachFolder) → Integer

 MSG GetPrefs (stripLineFeed; msgFolder; attachFolder) → Integer

 MSG FindHeader (fileName; headerLabel; headerValue) → Integer

 MSG MessageSize (fileName; headerSize; bodySize; msgSize) → Integer

 MSG GetHeaders (fileName; offset; length; headerText) → Integer

 MSG GetMessage (fileName; offset; length; todyText) → Longit

 MSG GetMessage (fileName; offset; length; rawText) → Integer

 MSG HasAttach (fileName; attachCount) → Integer

 MSG Mestarct (fileName; decode; attachmentPath; enclosureList) → Integer

 MSG Delete (fileName(; folder)) → Integer

 MSG Charset (decodeHeaders; bodyCharset) → Integer

	comma	

- IMAP_SetPrefs (stripLineFeed; msgFolder) Integer -- Theme: IC IMAP Review Mail
- IMAP_Download (imap_ID; msgNum; headerOnly; fileName{; updateSeen}) Integer -- Theme: <u>IC IMAP Review Mail</u>
- Integer -- Theme: IC POP3 Review Mail POP3_SetPrefs (stripLineFeed; msgFolder{; attachFolder})
- POP3_DownLoad (pop3_ID; msgNumber; headerOnly; fileName) Integer -- Theme: IC POP3 Review Mail
- POP3_Charset (decodeHeaders; bodyCharset) Integer -- Theme: IC POP3 Review Mail
- <u>SMTP_Charset</u> (encodeHeaders; bodyCharset) Integer -- Theme: <u>IC Send Mail</u>

Downloading Mail, Overview

version 6.8.1 (Modified)

The set of commands prefixed by "MSG_" allows the user to manipulate mail messages which have been saved as local files using the <u>POP3_Download</u> or <u>IMAP_Download</u> 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_SetPrefs (stripLineFeed; msgFolder; attachFolder) -> Integer

Parameter	Type		Description
stripLineFeed	Integer	\rightarrow	0 = Do not strip LineFeeds, 1 = Strip LineFeeds,
			-1 = No Change
msgFolder	Text	\rightarrow	Messages folder path ("" = no change)
attachFolder	Text	\rightarrow	Attachments folder path ("" = no change)

Function result Integer \leftarrow Error code

Description

The MSG_SetPrefs command sets the preferences for all MSG commands.

stripLineFeed is an integer value specifying how LineFeed characters will be handled in downloaded messages. Most Internet messages 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 the message "as is". A value of 1 will strip linefeed characters from messages. A value of -1 will leave this preference as it has been set previously. The default option defaults to 1 and will automatically strip any linefeeds found in messages.

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

Compatibility note (version 6.8.1): If the MSG_SetPrefs command is not used, the stripLineFeed and msgFolder parameters of the POP3 SetPrefs command will be taken into account if this command has been used previously. If the MSG_SetPrefs command is used, the POP3 SetPrefs parameters are ignored.

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.

Compatibility note (version 6.8.1): This parameter is also found in POP3_SetPrefs and MSG_SetPrefs; therefore, you can modify it using either of these two commands.

We strongly recommend that you use the MSG_SetPrefs command. The POP3_SetPrefs parameter is used for compatibility reasons and will no longer be used in the future. The attachFolder parameter of the POP3_SetPrefs command is optional; therefore, we recommend that you do not use this parameter. This recommendation also applies to POP3_GetPrefs.

See Also

IMAP_Download, IMAP_SetPrefs, MSG_Extract, POP3_DownLoad, POP3_SetPrefs.

 $MSG_GetPrefs \ (stripLineFeed; msgFolder; attachFolder) \longrightarrow Integer$

Parameter	Type		Description
stripLineFeed	Integer	\leftarrow	0 = Do not strip CR/LF, 1 = Strip CR/LF
msgFolder	Text	\leftarrow	Messages folder path ("" = no change)
attachFolder	Text	\leftarrow	Attachments folder path ("" = no change)

Function result Integer \leftarrow Error code

Description

The $\mbox{\it MSG_GetPrefs}$ command returns the current preferences for the MSG commands.

The preferences are returned into the variables listed in the parameters.

 $\mathit{stripLineFeed}$ returns the current setting of the user's preference for linefeed stripping.

 ${\it msgFolder} \ {\rm is\ a\ text\ variable\ which\ returns\ the\ local\ path name\ to\ the\ default\ folder\ in\ which\ retrieved\ messages\ are\ stored.}$

 $\it attach Folder is a text variable which returns the local pathname to the default folder in which extracted attachments are stored$

See Also

MSG_SetPrefs, POP3_SetPrefs.

MSG FindHeader

version 6.8.1 (Modified)

MSG_FindHeader (fileName; headerLabel; headerValue) -> Integer

Parameter	Туре		Description
fileName	Text	\rightarrow	Filename (path defaults to message folder)
headerLabel	String	\rightarrow	Header label ("From:", "To:", "Subject:", etc.)
headerValue	Text	\leftarrow	Value

Function result Integer ← Error Code

Description

Given the fileName of a message document retrieved to disk by the POP3_Download or IMAP_Download command, the MSG_FindHeader command 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 <u>POP3_SetPrefs</u> or <u>MSG_SetPrefs</u> (see Compatibility note). If no folder has been specified by <u>POP3_SetPrefs</u>, the path will default to the folder containing the structure of the database (with 4D single-user) or to the 4D Client folder (with 4D Server).

Compatibility note (version 6.8.1): If the MSG_SetPrefs command is not used, the POP3_SetPrefs msgFolder parameter will be used; if MSG_SetPrefs is used, the POP3_SetPrefs msgFolder parameter is ignored.

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 where the command will return the value assigned to the specified header field. Since the headerValue parameter can include extended characters, you can automate their management using the POP3 Charset or MSG Charset command.

Compatibility note (version 6.8.1): If the MSG_Charset command is not used, the POP3_Charset bodyCharset parameter will be used; if MSG_Charset is used, the POP3_Charset bodyCharset parameter is ignored.

See Also

 $\underline{IMAP_Download}, \underline{MSG_Charset}, \underline{MSG_SetPrefs}, \underline{POP3_Charset}, \underline{POP3_DownLoad}, \underline{POP3_SetPrefs}.$

MSG_MessageSize

version 6.8.1 (Modified)

 $MSG_MessageSize \ (fileName; headerSize; bodySize; msgSize) \longrightarrow Integer$

Parameter	Type		Description
fileName	Text	\rightarrow	Filename (path defaults to message folder)
headerSize	Longint	\leftarrow	Header size (subtracts linefeeds if Prefs ON)
bodySize	Longint	\leftarrow	Body size (subtracts linefeeds if Prefs ON)
msgSize	Longint	\leftarrow	Entire message or file size (ignores Prefs)

Function result Integer ← Error Code

Description

Given the fileName of a message document retrieved to disk by the POP3_Download command, the MSG_MessageSize command 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 or MSG SetPrefs (see Compatibility note). If no folder has been specified by POP3 SetPrefs, the path will default to the folder containing the structure of the database (with 4D single-user) or to the 4D Client folder (with 4D Server).

 $\ensuremath{\textit{headerSize}}$ 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.

These two parameters take the *stripLineFeed* parameter set by <u>POP3_SetPrefs</u> or <u>MSG_SetPrefs</u> into account.

Compatibility note (version 6.8.1): If the MSG_SetPrefs command is not used, the POP3_SetPrefs msgFolder and stripLineFeed parameters will be taken into account if the POP3_SetPrefs command has been used previously; if MSG_SetPrefs is used, the POP3_SetPrefs msgFolder and stripLineFeed parameters are ignored.

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

See Also

 $\underline{IMAP_Download}, \underline{MSG_SetPrefs}, \underline{POP3_DownLoad}, \underline{POP3_SetPrefs}.$

MSG GetHeaders

version 6.8.1 (Modified)

MSG_GetHeaders (fileName; offset; length; headerText) -> Integer

Parameter	Type		Description
fileName	Text	\rightarrow	Filename (path defaults to message folder)
offset	Longint	\rightarrow	Starting offset into headers (0 = start of header)
length	Longint	\rightarrow	Number of characters
headerText	Text	\leftarrow	Header text (removes linefeeds if Prefs ON)

Function result Integer \leftarrow Error Code

Description

The MSG_GetHeaders command 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 or MSG SetPrefs (see Compatibility note). If no folder has been specified, the path will default to the folder containing the structure of the database (with 4D single-user) or to the 4D Client folder (with 4D Server).

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 <u>MSG_MessageSize</u>.

 $\textit{headerText} \ \text{receives the text of the header. This parameter takes the \textit{stripLineFeed} \ parameter set by \underline{\textit{POP3_SetPrefs}} \ \text{or} \ \underline{\textit{MSG_SetPrefs}} \ \text{into account.}$

Compatibility note (version 6.8.1): If the <u>MSG_SetPrefs</u> command is not used, the <u>POP3_SetPrefs</u> msgFolder parameter will be used; if <u>MSG_SetPrefs</u> is used, the <u>POP3_SetPrefs</u> msgFolder parameter is ignored.

See Also

MSG_MessageSize, MSG_SetPrefs, POP3_SetPrefs.

MSG GetBody

version 6.8.1 (Modified)

 $MSG_GetBody \ (fileName; \ offset; length; bodyText) {\:\longrightarrow\:} Longint$

Parameter	Type		Description
fileName	Text	\rightarrow	Filename (path defaults to message folder)
offset	Longint	\rightarrow	Starting offset into message body (0 = start of body)
length	Longint	\rightarrow	Number of characters
bodyText	Text	\leftarrow	Body text (removes linefeeds if Prefs ON)

Function result Longint \leftarrow Error Code

Description

The MSG_GetBody command 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 or MSG_SetPrefs (see Compatibility notes). If no folder has been specified by POP3_SetPrefs, the path will default to the folder containing the structure of the database (with 4D single-user) or to the 4D Client folder (with 4D Server).

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

length is the number of characters to return.

body/Ext receives the text of the message body. Since the body/Ext parameter can include extended characters, you can automate their management using the POP3 Charset or MSG Charset command (see Compatibility notes). This parameter takes the stripLineFeed parameter set by POP3 SetPrefs or MSG SetPrefs into account (see Compatibility notes).

Compatibility notes (version 6.8.1):

- If the <u>MSG_SetPrefs</u> command is not used, the <u>POP3_SetPrefs</u> msgFolder and stripLineFeed parameters will be taken into account. If <u>MSG_SetPrefs</u> is used, the <u>POP3_SetPrefs</u> msgFolder and stripLineFeed parameters are ignored.
- If the MSG_Charset command is not used, the POP3_Charset bodyCharset parameter will be used; if MSG_Charset is used, the POP3_Charset bodyCharset parameter is ignored.

See Also

MSG_Charset, MSG_SetPrefs, POP3_Charset, POP3_SetPrefs.

MSG_GetMessage

version 6.8.1 (Modified)

 $MSG_GetMessage \ (fileName; \ offset; \ length; \ rawText) \longrightarrow Integer$

Parameter	Type		Description
fileName	Text	\rightarrow	Filename (path defaults to message folder)
offset	Longint	\rightarrow	Starting offset into message file (0 = start of file)
length	Longint	\rightarrow	Number of characters
rawText	Text	\leftarrow	Raw text (ignores Prefs)

Function result Integer \leftarrow Error Code

Description

The MSG_GetMessage command 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 or MSG_SetPrefs (see Compatibility note). If no folder has been specified, the path will default to the folder containing the structure of the database.

Compatibility note (version 6.8.1): If the MSG_SetPrefs command is not used, the POP3_SetPrefs msgFolder parameter will be used; if MSG_SetPrefs is used, the POP3_SetPrefs msgFolder parameter is ignored.

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

 $\ensuremath{\textit{length}}$ 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

 $\underline{MSG_SetPrefs}, \underline{POP3_SetPrefs}.$

MSG_HasAttach (fileName; attachCount) -> Integer

Parameter	Type
fileName	Text
attachCount	Integer

Description

- → Filename (path defaults to message folder)
- ← Count of Attachments

Function result Integer \leftarrow Error Code

Description

If the file has attachments, the MSG_HasAttach command 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 or MSG SetPrefs (see Compatibility note). If no folder has been specified, the path will default to the folder containing the structure of the database (with 4D single-user) or to the 4D Client folder (with 4D Server).

Compatibility note (version 6.8.1): If the MSG_SetPrefs command is not used, the POP3_SetPrefs msgFolder parameter is used; if MSG_SetPrefs is used, the POP3_SetPrefs msgFolder parameter is ignored.

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

See Also

MSG_SetPrefs, POP3_SetPrefs.

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

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

Function result Integer ← Error Code

Description

The MSG_Extract command 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 or MSG SetPrefs (see Compatibility note). If no folder has been specified, the path will default to the folder containing the structure of the database (with 4D single-user) or to the 4D Client folder (with 4D Server).

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 or MSG_SetPrefs (see Compatibility note). If no FolderPath has been specified, the attachment will be saved in the same folder as the database structure.

Compatibility note (version 6.8.1): If the MSG_SetPrefs command is not used, the POP3_SetPrefs msgFolder and attachmentPath parameters are used; if MSG_SetPrefs is used, the POP3_SetPrefs msgFolder and attachmentPath parameters are ignored.

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.

See Also

MSG_SetPrefs, POP3_SetPrefs.

 $MSG_Delete\ (fileName\{;\ folder\}) {\:\longrightarrow\:} Integer$

Parameter	Type		Description
fileName	Text	\rightarrow	Filename (path defaults to message folder)
folder	Integer	\rightarrow	0 = Message Folder, 1 = Attachment Folder

Function result Integer \leftarrow Error Code

Description

The MSG_Delete command 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 folder parameter is taken into account based on the following:

- folder = 0: the file resides in the message folder specified by <u>POP3_SetPrefs</u> or <u>MSG_SetPrefs</u>.
- folder = 1: the file resides in the attachment folder specified by <u>POP3_SetPrefs</u> or <u>MSG_SetPrefs</u>

In both cases, if no folder is defined by <u>POP3_SetPrefs</u> or <u>MSG_SetPrefs</u>, the path will default to the folder containing the structure of the database (with 4D single-user) or in the 4D Client folder (with 4D Server).

Compatibility note (version 6.8.1): If the MSG_SetPrefs command is not used, the POP3_SetPrefs msgFolder and attachFolder parameters are used; if MSG_SetPrefs is used, the POP3_SetPrefs parameters are ignored.

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

See Also

MSG_SetPrefs, POP3_SetPrefs.

MSG Charset (decodeHeaders; bodyCharset) → Integer

Parameter **Type**

Description

decodeHeaders Integer

- -1 = Use current settings, 0 = Do not manage, 1 = Convert using the Mac OS char set if ISO-8859-1 or ISO-2022-JP, decode extended characters
- bodyCharset Integer
- -1 = Use current settings, 0 = Do not manage, 1 = Convert using the Mac OS char set if ISO-8859-1 or ISO-2022-JP

Function result Integer \leftarrow Error code

Description

The MSG_Charset command allows automatic support of messages containing extended characters during processing with the MSG commands. If this command is not called or has parameters set to 0, 4D Internet Commands version 6.8.1 or higher will work the same way as version 6.5.x

MSG_Charset allows, first of all, the setting of whether the extended characters header decoding must be managed and, second, the determining of whether the message body and header character set conversion must be managed.

This command is particularly useful for supporting extended characters included in message headers such as the "Subject" or electronic mail addresses (for instance, to decode an address such as "=?ISO-8859-1?Q?Test=E9?= <test@n.net >").

The decodeHeaders parameter specifies how to manage header decoding and conversion while executing MSG_FindHeader. Default value is set to 0.

- -1: Use current settings;
- 0: Do not manage:
- 1: Headers are decoded if necessary. If decoded, and if the specified character set is ISO-8859-1 or ISO-2022-JP, headers are converted using Mac OS ASCII code or Shift-JIS respectively.

The bodyCharset parameter specifies how to manage message body character set conversion while executing the MSG_GetBody command. Default value is set to 0.

- · -1: Use current settings;
- · 0: Do not manage;
- 1: If the "Body-Content-Type" character set is set to ISO-8859-1 or ISO-2022-JP, the message body is converted using Mac OS ASCII code or Shift-JIS respectively.

Compatibility note (version 6.8.1): If the MSG_Charset command is not used and the POP3_Charset command has been used, the MSG_FindHeader and MSG_GetBody commands will take the POP3_Charset parameters into account. If MSG_Charset is used, the POP3_Charset parameters are ignored.

Examples

1. Using version 6.5.x of 4D Internet Commands:

```
$Err:=MSG_FindHeader($msgfile; "From"; $from)
$from:=ISO to Mac($from)
$Err:=MSG FindHeader($msgfile; "To";$to)
to:= ISO to Mac(to)
$Err:=MSG_FindHeader($msgfile; "Cc"; $cc)
c:= ISO to Mac($cc)
$Err:=MSG_FindHeader($msgfile; "Subject"; $subject)
$subject:=ISO to Mac($subject)
$Err:=MSG MessageSize($msgfile;$HdrSize;$BdySize;$msgSize)
$Err:=MSG GetBody($msgfile;0;$BdySize;$BodyContent)
$BodyContent:=<u>ISO to Mac</u>($BodyContent)
```

2. Using version 6.8.1 or higher of 4D Internet Commands:

```
$Err:=MSG_Charset(1;1)
$Err:=MSG_FindHeader($msgfile;"From";$from)
$Err:=MSG FindHeader($msgfile;"To";$to)
$Err:=MSG FindHeader($msgfile;"Cc";$cc)
$Err:=MSG_FindHeader($msgfile; "Subject"; $subject)
$Err:=<u>MSG MessageSize</u>($msgfile;$HdrSize;$BdySize;$msgSize)
$Err:=MSG GetBody($msgfile;0;$BdySize;$BodyContent).
```



IC File Transfer

- File Transfer, Overview
 FTP Progress (left; top; windowTitle; thermoText; cancel) → Integer
 FTP Login (hostName; userName; password; ftp_ID{; welcomeText}) → Integer
 FTP GetDirList (ftp_ID; directory; names; sizes; kinds; modDates{; modTimes}) → Integer
 FTP ChangeDir (ftp_ID; directory) → Integer
 FTP PrintDir (ftp_ID), directory) → Integer
 FTP GetFileInfo (ftp_ID; hostPath; size; modDate) → Integer
 FTP MakeDir (ftp_ID) → Integer
 FTP RemoveDir (ftp_ID; directory) → Integer
 FTP RemoveDir (ftp_ID; hostPath; newPathName) → Integer
 FTP MacBinary (ftp_ID; mostPath; newPathName) → Integer
 FTP MacBinary (ftp_ID; macBinaryMode) → Integer
 FTP Append (ftp_ID; localPath; hostPath; progress) → Integer
 FTP GetFype (ftp_ID; ftp_Mode) → Integer
 FTP SetIvpe (ftp_ID; ftp_Mode) → Integer
 FTP System (ftp_ID; systemInfo) → Integer
 FTP Remove (ftp_ID; hostPath; localPath; progress) → Integer
 FTP Remove (ftp_ID; hostPath; localPath; progress) → Integer
 FTP System (ftp_ID; passiveMode) → Integer
 FTP Logout (ftp_ID) → Integer
 FTP SetPassive (ftpID; passiveMode) → Integer
 FTP GetPassive (ftpID; passiveMode) → Integer
 FTP GetPassive (ftpID; passiveMode) → Integer

Other related commands:

 IT Decode (fileName; decodedFile; decodeMode) Integer -- Theme: IC Utilities

File Transfer, Overview

version 2003 (Modified)

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.

Notes

- 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.
- For greater flexibility, 4D Internet commands let you pass a POP3, IMAP or FTP connection reference directly to low-level TCP commands and vice versa. For more information, refer to the <u>Low Level Routines</u>, <u>Overview</u> section

version 6.5

FTP Progress (left; top; windowTitle; thermoText; cancel) -> Integer

Parameter	Type		Description
left	Integer	\rightarrow	Left window coordinate
top	Integer	\rightarrow	Top window coordinate
windowTitle	String	\rightarrow	Thermometer Window Title
thermoText	String	\rightarrow	Text above thermometer progress
cancel	String	\rightarrow	Cancel button text

Function result Integer ← Error Code

Description

The FTP_Progress command defines window coordinates and dialog box text for the FTP progress indicator. The progress indicator can appear during calls to FTP_Send, FTP_Append 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, FTP_Append 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.

window Title is the title of the thermometer progress window. In the following example, the window title is "Getting '/pub/CGMiniViewer.hqx'" If window Title 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".



Example

```
$error:=FTP_Progress (-1;-1;"Getting '/pub/CGMiniViewer.hqx'";"*";"*")
Case of
    : (FTP_Login ("ftp.4d.com";"anonymous";"dbody@aol.com";vFTP_ID;vFTP_Msg)#0)
    : (FTP_Receive (vFTP_ID;"/pub/CGMiniViewer.hqx";"HardDrive:Docsf:4D";1)#0)
    : (FTP_Logout (vFTP_ID)#0)
Else
    $OK:=True    `all commands executed without error
End case
```

Note: For more information about this particular use of the Case of structure, please refer to Appendix A, Programming Tips.

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

Parameter	Type		Description
hostName	String	\rightarrow	Host name or IP address
userName	String	\rightarrow	User name
password	String	\rightarrow	Password
ftp_ID	Longint	\leftarrow	Reference to this new FTP session
welcomeText	Text	\leftarrow	FTP Welcome text

Function result Integer Error Code

Description

The FTP_Login command 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.

welcome Text 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

Note: For more information about this particular use of the *Case of* structure, please refer to <u>Appendix A, Programming Tips</u>.

See Also

FTP_Logout.

 $FTP_GetDirList\ (ftp_ID;\ directory;\ names;\ sizes;\ kinds;\ modDates\{;\ modTimes\}) \xrightarrow{} Integer$

Parameter	Type		Description
ftp_ID	Longint	\rightarrow	Reference to a FTP login
directory	Text	\rightarrow	Unix directory pathname
		\leftarrow	Current directory
names	Str Txt Array	\leftarrow	List of Names
sizes	Longint Array	←	List of Sizes
kinds	Integer Array	←	List of 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	<i>r</i> ←	List of Modification Dates
modTimes	Longint Array	\leftarrow	List of Modification Times

Function result Integer \leftarrow Error Code

Description

The FTP_GetDirList command will retrieve a list of the objects in a directory of the FTP session identified by ftp_ID. Information on the names, sizes, types, modification dates and, optionally, modification times of the directory items is returned in 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.

 $\mathit{ftp_ID}$ is the long integer reference to the FTP session established with $\mathit{\underline{FTP_Login}}$.

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 list 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

block-type special file
character-type special file
symbolic link (aliases on files or folders)
FIFO special file
AF_UNIX address family socket

Note: In the case of a symbolic link (kind=4), the FTP server returns a particular pathname (Alias name + symbol + pathname to the source file or folder). If you try to use this pathname to access source files or folders, an error will be returned. You MUST extract the pathname to the source file or folder from the string returned by FTP_GetDirList which starts just after the symbolic character. Otherwise, commands such as FTP_GetFileInfo will return the error –10085 since the file or folder will not be found.

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

modTimes is a longint array that receives the time of the last modification for each object in the directory.

Reminder: In 4th Dimension, longint arrays are used for handling time type data (each array item represents a number of seconds). Use the <u>Time string</u> command to convert these values into the HH:MM:SS format.

See Als

 $\underline{FTP_ChangeDir}, \underline{FTP_Login}, \underline{FTP_PrintDir}, \underline{FTP_VerifyID}.$

version 6.7

 $FTP_ChangeDir\ (ftp_ID;\ directory) {\:\longrightarrow\:} Integer$

Parameter	Type		Description
ftp_ID	Longint	\rightarrow	Reference to a FTP login
directory	Text	\rightarrow	Unix directory pathname

Function result Integer \leftarrow Error Code

Description

The FTP_ChangeDir command changes your current working directory (CWD) to the path given to the directory parameter.

The commands <u>FTP_GetDirList</u> and <u>FTP_GetFileInfo</u> will also change the current working directory. However, executing the <u>FTP_ChangeDir</u> command is faster and needs less parameters.

 ftp_ID is the long integer reference to the FTP session established with $\underline{FTP_Login}$.

directory is a text value in the format of a HostPath which references an existing FTP directory. An error will be returned if the directory does not exist or if you do not have sufficient access priveleges to perform this action. In this case, the current working directory will be left unchanged.

Example

This statement will set the CWD to the FTP root:

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

See Also

FTP_GetDirList, FTP_GetFileInfo, FTP_PrintDir.

 $FTP_PrintDir\ (ftp_ID;\ directory) \longrightarrow Integer$

Parameter	Туре		Description
ftp_ID	Longint	\rightarrow	Reference to a FTP login
directory	Text	\leftarrow	Unix directory pathname

Function result Integer \leftarrow Error Code

Description

The FTP_PrintDir command returns your current working directory (CWD) in the *directory* parameter.

The command FTP_GetDirList will also return the current working directory. However, executing the FTP_PrintDir command is faster and needs less parameters.

 ftp_ID is the long integer reference to the FTP session established with $\underline{FTP_Login}$.

The directory parameter returns the current working directory (CWD).

Example

This example will return the current working directory in the \$Cwd variable.

\$err:=FTP_PrintDir(ftp_ID;\$Cwd)

See Also

FTP_GetDirList.

FTP_GetFileInfo

version 6.5

 $FTP_GetFileInfo\ (ftp_ID;\ hostPath;\ size;\ modDate) \longrightarrow Integer$

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 \leftarrow Error Code

Description

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

 ftp_ID is the long integer reference to the FTP session established with $\underline{FTP_Login}$.

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

Note: The FTP_GetFileInfo command may modify the Current Working Directory (CWD) if hostPath is a full pathname which indicates a directory different from the CWD. In this case, the CWD becomes the directory defined by the hostPath parameter.

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

version 6.5

 $FTP_VerifyID (ftp_ID) \longrightarrow Integer$

Parameter	Type
ftp_ID	Longin

Description

- → Reference to a FTP login
- \leftarrow 0 = Connection has already closed

Function result Integer \leftarrow 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.

Son Alco

FTP_Login.

FTP_MakeDir

version 6.5

 $FTP_MakeDir (ftp_ID; directory) \longrightarrow Integer$

Parameter	Type	Description
ftp_ID	Longint →	Reference to a FTP login
directory	Text →	Unix directory pathname

Function result Integer \leftarrow Error Code

Description

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

 ftp_ID is the long integer reference to the FTP session established with $\underline{FTP_Login}$.

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.

Note: You can change the CWD using the FTP ChangeDir command. You can also determine the CWD at any time using the FTP PrintDir command.

See Also

 $\underline{FTP_ChangeDir}, \underline{FTP_GetDirList}, \underline{FTP_PrintDir}, \underline{FTP_RemoveDir}.$

FTP_RemoveDir

version 6.5

 $FTP_RemoveDir\ (ftp_ID;\ directory) \longrightarrow Integer$

Parameter	Type		Description
ftp_ID	Longint	\rightarrow	Reference to a FTP login
directory	Text	\rightarrow	Unix directory pathname

Function result Integer \leftarrow Error Code

Description

Given an acceptable *directory* name, the FTP_RemoveDir command will delete a Folder 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 $\underline{FTP_Login}$.

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.

Note: You can change the CWD using the FTP_ChangeDir command. You can also know the CWD at any time using the FTP_PrintDir command.

See Also

 $\underline{FTP_ChangeDir}, \underline{FTP_GetDirList}, \underline{FTP_MakeDir}, \underline{FTP_PrintDir}.$

FTP_Rename

version 6.5

 $FTP_Rename~(ftp_ID;~hostPath;~newPathName) \longrightarrow Integer$

Parameter	Type		Description
ftp_ID	Longint	\rightarrow	Reference to a FTP login
hostPath	Text	\rightarrow	Pathname to document on FTP Server
newPathName	Text	\rightarrow	New document name

Function result Integer \leftarrow Error Code

Description

Given a pathname to a file in the format of a hostPath, the FTP_Rename command 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 $\underline{\textit{FTP_Login}}$.

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.

 $\ensuremath{\textit{newPathName}}$ contains the value you wish to rename the remote document.

Note: You can change the CWD using the FTP_ChangeDir command. You can also know the CWD at any time using the FTP_PrintDir command.

See Also

 $\underline{FTP_ChangeDir},\,\underline{FTP_PrintDir}.$

FTP_Delete

version 6.5

 $FTP_Delete~(ftp_ID;~hostPath) \longrightarrow Integer$

Parameter	Туре		Description
ftp_ID	Longint	\rightarrow	Reference to a FTP login
hostPath	Text	\rightarrow	Pathname to document

Function result Integer \leftarrow Error Code

Description

Given a pathname to a file in the format of a HostPath, the FTP_Delete command 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 $\underline{FTP_Login}$.

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.

Note: You can change the CWD using the FTP_ChangeDir command. You can also know the CWD at any time using the FTP_PrintDir command.

See Also

 $\underline{FTP_ChangeDir}, \underline{FTP_PrintDir}, \underline{FTP_RemoveDir}.$

FTP_MacBinary (ftp_ID; macBinaryMode) -> Integer

ParameterTypeDescriptionftp_IDLongint→ Reference to a FTP loginmacBinaryMode Integer→ -1 = Get Current setting, 1 = Enable, 0 =

← Current setting (if -1 passed)

Function result Integer \leftarrow Error Code

Description

The FTP_MacBinary command 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.

Disable

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).

Warning: Not all FTP servers support the MacBinary protocol, in this case the error 10053 is returned at each FTP_MacBinary command call, whatever the value of the macBinaryMode parameter. Previously described behaviours become false.

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.

```
vUseMacBin:=-1
$error:=FTP_MacBinary (vFTP_ID;vUseMacBin)
If($error=10053)
  MacBinaryIsSupported:=False `Ftp server doesn't support the MacBinary protocol
  MacBinaryIsSupported:=<u>True</u>
End if
vLocalFile:=""
If(MacBinaryIsSupported)
  vUseMacBin:=1
  $error:=FTP_MacBinary (vFTP_ID;vUseMacBin) `Try to turn MacBinary on for the downloa
$error:=FTP_Receive (vFTP_ID;"MyApplication";vLocalFile;cbShowTherm)
If ($error=0) & (vUseMacBin=1) `If received OK and the file is in MacBinary format
  vDecodePath:=""
  DELETE DOCUMENT(vLocalFile) `If sucessful decode of source, then delete it.
  End if
End if
```

See Also

IT_Decode

FTP Send (ftp ID; localPath; hostPath; progress) -> Integer

Parameter	Type		Description
ftp_ID	Longint	\rightarrow	Reference to a FTP login
localPath	Text	\rightarrow	Pathname of document to send
hostPath	Text	\rightarrow	Pathname to destination of document
progress	Integer	\rightarrow	1 = Show Progress, 0 = Hide Progress

Function result Integer \leftarrow Error Code

Description

Given a reference to an open FTP session, the pathname of a document to send and the destination pathname, the FTP_Send command 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 (with 4D single-user) or in the 4D Client folder (with 4D Server) 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 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 <u>Glossary and Terminology</u> 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.4d.com";"anonymous";vEmailID;vFTP_ID;vFTP_Msg)#0)
     : (FTP_Progress (-1;-1;"Progress window";"Getting requested file...";"Cancel")#0)
     : (FTP_Send (vFTP_ID;"My Hard Drive:Documents:July Sales Report";"/pub/reports/";1)#(
     : (FTP_Logout (vFTP_ID)#0)
Else
     $OK:=True     `all commands executed without error
End case
```

Note: For more information about this particular use of the Case of structure, please refer to Appendix A, Programming Tips.

Example 2

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

See Also

FTP_Progress, FTP_Receive.

FTP_Append

version 6.5

 $FTP_Append (ftp_ID; localPath; hostPath; progress) \mathop{\longrightarrow}\limits Integer$

Parameter	Type		Description
ftp_ID	Longint	\rightarrow	Reference to a FTP login
localPath	Text	\rightarrow	Pathname of document to send
hostPath	Text	\rightarrow	Pathname to destination of document
progress	Integer	\rightarrow	1 = Show Progress, 0 = Hide Progress

Function result Integer ← Error Code

Description

The FTP_Append command 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

version 6.5

 $FTP_GetType~(ftp_ID;~ftp_Mode) \longrightarrow Integer$

Parameter	Type		Description
ftp_ID	Longint	\rightarrow	Reference to a FTP login
ftp_Mode	String	\leftarrow	"A" = Ascii; "I" = Image; "L 8" = Logical 8-bit

Function result Integer \leftarrow Error Code

Description

The FTP_GetType command 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

version 6.5

 $FTP_SetType\ (ftp_ID;\ ftp_Mode) {\:\longrightarrow\:} Integer$

Parameter	Type		Description
ftp_ID	Longint	\rightarrow	Reference to a FTP login
ftp_Mode	String	\rightarrow	"A" = Ascii; "I" = [Default] Image; "L 8" = Logical 8-bit

Function result Integer \leftarrow Error Code

Description

The FTP_SetType command 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.

 $\mathit{ftp_ID}$ is the long integer reference to the FTP session established with $\mathit{\underline{FTP_Login}}$.

ftp_Mode should contain a code as described above indicating the preferred transfer mode to use for future Send/Receive operations. By default, the Image ("I") tranfer mode is used.

See Also

FTP_GetType.

 $FTP_System~(ftp_ID;~systemInfo) \longrightarrow Integer$

Parameter	Type		Description
ftp_ID	Longint	\rightarrow	Reference to a FTP login
systemInfo	String	\leftarrow	System Information

Function result Integer \leftarrow Error Code

Description

The FTP_System command 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.

 ${\it systemInfo} \ {\it will} \ {\it contain} \ {\it information} \ {\it about} \ {\it the} \ {\it FTP} \ {\it server}.$

FTP_Receive (ftp_ID; hostPath; localPath; progress) -> Integer

Parameter	Type		Description
ftp_ID	Longint	\rightarrow	Reference to a FTP login
hostPath	Text	\rightarrow	Pathname of document to receive
localPath	Text	\rightarrow	Pathname to destination of document
		\leftarrow	Resulting file pathname (if "" passed)
progress	Integer	\rightarrow	0 = Hide Progress, 1 = Show Progress

Function result Integer ← Error Code

Description

The FTP_Receive command 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 <u>Glossary and Terminology</u> 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 (with 4D single-user) or the 4D Client folder (with 4D Server). 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

```
vUseMacBin:=-1
$error:=FTP_MacBinary (vFTP_ID;vUseMacBin)
If($error=10053)
  MacBinaryIsSupported:=False `Ftp server doesn't support the MacBinary protocol
Else
  MacBinaryIsSupported:=True
End if
vLocalFile:=""
If(MacBinaryIsSupported)
  vUseMacBin:=1
  $error:=FTP_MacBinary (vFTP_ID;vUseMacBin) `Try to turn MacBinary on for the downloa
  $error:=FTP_Receive (vFTP_ID; "CGMiniViewer.hqx"; vLocalFile; cbShowTherm)
  If ($error=0) & (vUseMacBin=1)
     vDecodePath:=""
     DELETE DOCUMENT(vLocalFile) `If successful decode of source, then delete it.
     End if
  End if
End if
```

See Also

FTP_MacBinary, IT_Decode.

FTP_Logout

version 6.5

 $FTP_Logout\ (ftp_ID) {\:\longrightarrow\:} Integer$

Parameter Type Description

ftp_ID Longint

→ Reference to a FTP login

← 0 = session successfully closed

Function result Integer \leftarrow Error Code

Description

Given a reference to an open FTP session, the FTP_Logout command 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

See Also

FTP_Login.

FTP_SetPassive

version 6.5.3 (Modified)

 $FTP_SetPassive (ftpID; passiveMode) \longrightarrow Integer$

Parameter	Type		Description
ftpID	Longint	\rightarrow	Reference to a FTP login
passiveMode	Integer	\rightarrow	0=Active mode, 1=Passive mode (default mode)

Function result Integer \leftarrow Error Code

Description

The FTP_SetPassive command sets the data stream transfer mode between an FTP Server and an FTP Client while executing commands such as FTP_GetDirList, FTP_Send, FTP_Append or FTP_Receive. Data stream transfer mode setting will be applied to these commands once the command FTP_SetPassive has been executed.

Exchanges between a FTP Server and a FTP Client are based upon two streams: the control stream (port 21 by default) and the data stream (port 20 by default). Usually, FTP Servers are defined as "active" since they open and manage the data connection.

For historical reasons, 4D Internet Commands default data stream transfer mode consists of asking the FTP Server to work in Passive mode. It means that prior to each exchange on the data stream, the FTP command "PASV" is sent.

However, some FTP Servers do not support the passive mode, and also firewalls may not allow it. In these cases, you will need to set the active mode as the current data stream transfer mode.

Note: You may need to check with the network administrator whether the active or the passive mode is used for FTP exchanges.

 ftp_ID is the long integer reference to the FTP session established with $\underline{FTP_Login}$.

The *passiveMode* parameter value specifies the data stream transfer mode:

- \bullet a value of 0 will specify the FTP Server to work in Active mode
- \bullet a value of 1 will specify the FTP Server to work in Passive mode (default value).

See Also

FTP_GetPassive.

 $FTP_GetPassive \ (ftp_ID; passiveMode) {\:\longrightarrow\:} Integer$

Parameter	Type		Description
ftp_ID	Longint	\rightarrow	Reference to a FTP login
passiveMode	Integer	\leftarrow	Current data stream transfer mode
			0=Active mode, 1=Passive mode

Function result Integer \leftarrow Error Code

Description

The $\ensuremath{\mathit{FTP_GetPassive}}$ command returns the current data stream transfer mode.

For more details about FTP transfer modes, refer to the $\underline{\textit{FTP_SetPassive}}$ command description.

 ftp_ID is the long integer reference to the FTP session established with $\underline{FTP_Login}$.

 $passive Mode\ {\it returns}\ the\ current\ data\ stream\ transfer\ mode:$

- \bullet if 0 is returned, the FTP Server is currently asked to work in Active mode.
- ullet if 1 is returned, the FTP Server is currently asked to work in Passive mode (default value).

See Also

FTP_SetPassive.

IC TCP/IP

- Low Level Routines, Overview
 TCP Open (hostName; remotePort; tcp_ID{; sessionSettings}) → Integer
 TCP Listen (remoteHost; localPort; remotePort; timeout; tcp_ID) → Integer
 TCP Send (tcp_ID; sendText) → Integer
 TCP Receive (tcp_ID; text) → Integer
 TCP SendBLOB (tcp_ID; blobToSend) → Integer
 TCP ReceiveBLOB (tcp_ID; blobToReceive) → Integer
 TCP State (tcp_ID; statusCode) → Integer
 TCP_Close (tcp_ID) → Integer

- Other related commands:

 Appendix B, TCP Port Numbers -- Theme: Appendixes
- IT_SetTimeOut (timeout) Integer -- Theme: IC Utilities

version 2003 (Modified)

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. TCP Open allows the use of SSL (Secured Socket Layer) protocol which permits a secured connection. 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.

In any case, any TCP connection opened must be closed subsequently using the TCP_Close command.

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.

Connection references in TCP commands

4D Internet commands allow the passing of POP3, IMAP or FTP connection references directly to low-level TCP commands and vice versa.

In fact, on the one hand, protocols are constantly evolving which leads to the creation of new commands; on the other, some software packages make their own interpretation of RFCs — rendering standardized implementations unusable. Using low-level TCP commands, developers can create the high-level functions they need themselves (instead of using existing functions or to fill in for a function that does not exist).

This significantly increases compatibility and development possibilities since developers can create their own high-level commands without having to rewrite all the commands needed for using a protocol.

In this example, the <u>IMAP_Capability</u> command is replaced by an equivalent function developed using TCP_IP commands.

• Here is the initial method using the IMAP Capability command: \$ErrorNum:=<u>IMAP_Login</u>(vHost;vUserName;vUserPassword;vImap_ID) If(\$ErrorNum=0) C TEXT(vCapability) \$ErrorNum:=<u>IMAP_Capability</u>(vImap_ID;vCapability) IMAP command using the vImap_ID parameter End if \$ErrorNum:=IMAP_Logout(vImap_ID) · This method can be replaced by: \$ErrorNum:=<u>IMAP_Login</u>(vHost;vUserName;vUserPassword;vImap_ID) If(\$ErrorNum = 0)C_TEXT(vCapability) TCP method using the value of the vImap_ID parameter: \$ErrorNum:=My_IMAP_Capability(vImap_ID) IMAP commands using the vImap_ID parameter End if \$ErrorNum:=IMAP_Logout(vImap_ID) • Here is the code of the My_IMAP_Capability function: C_LONGINT(\$1;\$vErrorNum;\$0) C_TEXT(\$vSentText;\$vReceivedText;vCapability) **C_TEXT**(\$2) \$vImap_Id:=\$1 \$vCmd_Id:="A001" ` This command ID must be unique (cf. RFC 2060) \$MyvtRequestCmd:="CAPABILITY" \$vSentText;:=\$vCmd_Id+""+\$MyvtRequestCmd+Character(13)+Character(10) \$vReceivedText:="" \$vErrorNum:=<u>TCP_Send(</u>\$vImap_Id;\$vSentText) If (\$vErrorNum=0) \$vErrorNum:=TCP_Receive(\$vImap_Id;\$vReceivedText) Case of `Reception error :(\$vErrorNum#0) vCapability:="" :(Position(\$vCmd_Id+" OK ";\$vReceivedText)#0) Command execution successful vCapability:=\$vReceivedText In this example, we do not process the string received :(Position(\$vCmd_Id+" BAD ";\$vReceivedText)#0) Failure of command execution (syntax error or unknown command) vCapability:=""

\$vErrorNum:=10096 End case End if \$0:=\$vErrorNum

TCP_Open (hostName; remotePort; tcp_ID{; sessionSettings}) → Integer

Parameter	Type		Description
hostName	String	\rightarrow	Host name or IP address
remotePort	Integer	\rightarrow	The remote port to connect to $(0 = any)$
tcp_ID	Longint	\leftarrow	Reference to this TCP session
$sessionSettings\ Integer \qquad \rightarrow$		\rightarrow	TCP session settings
			0 = Synchron (Default if omitted)
			1 = Asynchron
			2 = SSL In Use, Synchron
			3 = SSL In Use, Asynchron

Function result Integer \leftarrow Error Code

Description

The TCP_Open command 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_SetTimeOut.

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.

Note: After a call to TCP_Open (or TCP_Listen), remotePort may contain a negative value if the value passed to this parameter is above 32767. This will not disturb the connection. As a workaround, you can use an intermediate variable:

```
$v_ RemotePort:=v_ RemotePort
$err:=TCP_Open (v_ RemoteHostIPAdr;0;v_ SessionID)
```

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.

session Settings is an optional parameter of the Integer type giving the user the ability to choose the TCP session settings. Note that these settings are applied to each TCP command called during the session. Default value sets to 0 (Synchron, not SSL).

SSL (Secured Socket Layer) is a protocol that allows secured TCP communications (see the 4th Dimension reference for more information and for installation requirements).

Any TCP connection opened using the ${\it TCP_Open}$ command must be closed later using the ${\it TCP_Close}$ command.

Asynchron/Synchron

Asynchronous mode returns control to the 4D kernel immediately without waiting for the connection process to be finished (without waiting for the connection with the remote host to be established). Asynchronous mode is useful for people who do not want all the TCP commands to use 4th Dimension time.

Synchronous mode returns control to the 4D kernel (to other 4D processes) only when the connection process is finished (successfully or not).

- \bullet 0 = Synchron mode (Default mode, run as previous versions of 4D Internet Commands)
- 1 = Asynchron mode
- 2 = SSL In Use, Synchron. All TCP commands using the reference to this TCP session (tcp_ID) will run in Synchronous mode and use SSL protocol.
- 3 = SSL In Use, Asynchron. All TCP commands using the reference to this TCP session (tcp_ID) will run in Asynchronous mode and use SSL protocol.

Note: An error 10089 may be returned when passing 2 or 3 if an SSL connection cannot be opened (SLI library not found in the 4D Extensions folder).

Example

You want to connect to a Web site using Https; check that SLI is correctly installed and open a connection using the 443 port number:

```
$vError:=TCP_Open (hostName; 443; tcp_ID;2)
...
$vError:=TCP_Close (tcp_ID) `Don't forget to close the session
```

See Also

IT_SetTimeOut.



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

Parameter	Type		Description
remoteHost	String	\rightarrow	Host name or IP address
		\leftarrow	IP address is a variable containing a null string
localPort	Integer	\rightarrow	Local port number, 0 = find an unused port to use
		←	Used local port number (if 0 passed)
remotePort	Integer	\rightarrow	Port number to be listening
remoter ort	integer	,	Tort number to be fistening
timeout	Integer	\rightarrow	# of seconds to wait, $0 =$ wait forever
tcp_ID	Longint	\leftarrow	Reference to this TCP session

Function result Integer \leftarrow Error Code

Description

The TCP_Listen command 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.
- $If a \ variable \ containing \ a \ null \ string \ is \ passed \ in \ this \ parameter, \ it \ will \ return \ the \ IP \ address \ of \ the \ connected \ machine.$

Note: Under Windows, remoteHost will not accept an IP address from a distant machine; if this occurs, error –10049 "Specified address is not available from the local machine" will be generated. As a consequence, if you need to filter an IP address, it is better to use a variable containing a null string.

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.

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

Note: After a call to TCP_Listen (or TCP_Open), remotePort may contain a negative value if the value passed to this parameter is above 32767. This will not disturb the connection.

As a workaround, you can use an intermediate variable:

```
$v_ RemotePort:=v_ RemotePort
$err:=TCP_Listen (v_ RemoteHostIPAdr;0;$v_ RemotePort;30;v_ SessionID)
```

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.

Any TCP connection opened using the $\underline{\textit{TCP_Listen}}$ command must be closed later using the $\underline{\textit{TCP_Close}}$ command.

Example

See Also

Appendix B, TCP Port Numbers, TCP Open, TCP State



 $TCP_Send\ (tcp_ID; sendText) {\:\longrightarrow\:} Integer$

Parameter	Type		Description
tcp_ID	Longint	\rightarrow	Reference to an open TCP session

sendText \rightarrow Text to send

Function result Integer \leftarrow Error Code

Description

The TCP_Send command 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 <u>TCP_Open</u> or <u>TCP_Listen</u> command.

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

See Also

TCP_Listen, TCP_Open, TCP_SendBLOB.

TCP_Receive

version 6.5

TCP_Receive (tcp_ID; text) \longrightarrow Integer

Parameter	Type		Description
tcp_ID	Longint	\rightarrow	Reference to an open TCP session
text	Text	\leftarrow	Received Text

Function result Integer \leftarrow Error Code

Description

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

 tcp_ID is a long integer reference to an open TCP session as established with either the $\underline{TCP_Open}$ or $\underline{TCP_Listen}$ 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_SendBLOB.

TCP_SendBLOB

version 6.7

TCP_SendBLOB (tcp_ID; blobToSend) → Integer

Parameter	Type		Description
tcp_ID	Longint	\rightarrow	Reference to an open TCP session
blobToSend	BLOB	\rightarrow	Blob to send

Function result Integer \leftarrow Error Code

Description

The TCP_SendBLOB command sends data to the TCP session designated by tcp_ID. This command performs the same action as TCP_Send, except that it will send a BLOB instead of a text, which allows bypassing the 32K text limitation. Using this command, you can send binary objects.

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

blobToSend is the BLOB to be sent to the TCP session referenced by tcp_ID.

Note regarding Platform Independence: If you exchange BLOBs between Macintosh and PC platforms, it is up to you to manage byte swapping issues if necessary.

Example

This example will send a BLOB to the TCP session:

```
C_BLOB($Blob_Send)
C_TEXT(v_Txt_Send)
TEXT_TO_BLOB(v_Txt_Send;$Blob_Send; Text_without_length;*)
$err:=TCP_SendBLOB(v_tcp_ID;$Blob_Send)
```

See Also

TCP_Listen, TCP_Open, TCP_ReceiveBLOB, TCP_Send.

version 6.7

TCP_ReceiveBLOB (tcp_ID; blobToReceive) -> Integer

Parameter Type Description

 tcp_ID Longint \rightarrow Reference to an open TCP session

blobToReceive BLOB ← BLOB to receive data

Function result Integer ← Error Code

Description

Given a long integer reference to an established TCP session, the TCP_ReceiveBLOB command receives packets of data into blobToReceive.

This command performs the same action as TCP_Receive, except that it will receive data in a BLOB instead of a text, which allows bypassing the 32K text limitation. You can then receive binary objects.

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

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

Example

This example shows the typical structure of a method using TCP_ReceiveBLOB:

```
C_BLOB($Blob_Received;$Blob_All)
C_LONGINT($srcpos;$dstpos)
Repeat
    $Err:=TCP_ReceiveBLOB ($TCP_ID;$Blob_Received )
    $Err:=TCP_State ($TCP_ID;$State)
    $srcpos:=0
    $dstpos:= BLOB_size($Blob_All)
          `Concatenating received Blobs
          COPY_BLOB($Blob_Received;$Blob_All;$srcpos;$dstpos;BLOB_size($Blob_Received))
Until(($State=0) | ($Err#0))
```

See Also

TCP_Listen, TCP_Open, TCP_Receive, TCP_SendBLOB.

 $TCP_State (tcp_ID; statusCode) \rightarrow Integer$

Parameter	Type		Description
tcp_ID	Longint	\rightarrow	Reference to an open TCP
statusCode	Integer	\leftarrow	TCP status code

Function result Integer \leftarrow Error Code

Description

The TCP_State command returns an integer value corresponding to the state of a particular TCP connection.

 tcp_ID is a long integer reference to an open TCP session as established with either the $\underline{TCP_Open}$ or $\underline{TCP_Listen}$ 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 connections once they have performed their action, this example keeps receiving until the connection is dropped or an error occurs.

See Also

TCP_Listen, TCP_Open.

TCP_Close

version 6.5

 $TCP_Close\ (tcp_ID) \longrightarrow Integer$

Parameter Type tcp_ID Longint

Description

- → Reference to an open TCP session
- ← 0 = Session successfully closed

Function result Integer \leftarrow Error Code

Description

The *TCP_Close* command 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 <u>TCP_Open</u> or <u>TCP_Listen</u> 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.

IC UDP

- UDP Commands, Overview
 UDP New (localPort; udp_ID) → Integer
 UDP SendTo (udp_ID; hostName; remotePort; sendText) → Integer
 UDP ReceiveFrom (udp_ID; hostName; remotePort; text) → Integer
 UDP SendBLOBTo (udp_ID; hostName; remotePort; sendBlob) → Integer
 UDP ReceiveBLOBFrom (udp_ID; hostName; remotePort; blob) → Integer
 UDP Delete (udp_ID) → Integer

Other related commands:

<u>Low Level Routines, Overview</u> -- Theme: <u>IC TCP/IP</u>

vorcion 11

UDP (User Datagram Protocol) is an easy-to-implement protocol for sending data. It is faster and simpler than TCP (only 8 bytes of header as opposed to at least 20 bytes in TCP), but it does not offer the same level of reliability. It is useful for applications where data must arrive at their destination quickly. However, it does not allow verification of delivery, nor does it allow error-checking or recovery of data that was not delivered correctly.

Example

The following example illustrates how the list of 4D Servers running on a local network can be retrieved using UDP commands:

```
ARRAY STRING(255;asHost;0)
ARRAY STRING(32;asMachineName;0)
ARRAY STRING(32;asService;0)
ARRAY STRING(32;asDBName;0)
C_BLOB($Blob)
$Addr:="255.255.255.255"
$Port:=19813
$0ffset:=32
SET BLOB SIZE($Blob; 96; 0)
TEXT TO BLOB("4D Server"; $Blob; Mac text without length; $Offset)
$Err:=UDP_New(0;$udpID)
$Err:=UDP_SendBLOBTo($udpID;$Addr;$Port;$Blob)
$Secs:=2
$Timeout:=Milliseconds+($Secs*1000)
   DELAY PROCESS(Current process;6) `... in ticks
   SET BLOB SIZE($Blob;0;0)
   $PeerAddr:=$Addr
   $Err:=UDP_ReceiveBLOBFrom($udpID;$PeerAddr;$Port;$Blob)
   If(BLOB size($Blob)>0)
      $0ffset:=0
      $Host:=BLOB to text($Blob; Mac C string; $Offset; 32)
      $0ffset:=32
      $Service:=BLOB to text($Blob; Mac C string; $Offset; 32)
      $0ffset:=64
      $DBName:=BLOB to text($Blob, Mac C string;$0ffset;32)
      $Pos:=Find in array(asMachineName;$Host)
      If($Pos>0)
         APPEND TO ARRAY(asHost; $PeerAddr)
         APPEND TO ARRAY (asMachineName; $Host)
         APPEND TO ARRAY(asService;$Service)
         APPEND TO ARRAY(asDBName; $DBName)
      End if
   End if
Until(Milliseconds>$Timeout) | ($Err#0))
$Err:=UDP_Delete($udpID)
```

See Also

Low Level Routines, Overview.

 $UDP_New~(localPort;~udp_ID) \longrightarrow Integer$

Parameter	Type		Description
localPort	Integer	\rightarrow	Local port used for UDP socket
			(0 = find any unused port to use)
udp_ID	Longint	\leftarrow	UDP socket reference

Function result Integer \leftarrow Error code

Description

The <u>UDP_New</u> command can be used to create a UDP socket by passing the local port number to be used in the *localPort* parameter. If you pass 0, the command will find any unused port. The UDP socket reference will be returned in the *udp_ID* parameter. Once this socket is no longer needed, remember to close it using <u>UDP_Delete</u> in order to free up memory.

See Also

UDP_Delete.

UDP_SendTo

version 11

 $UDP_SendTo\ (udp_ID;\ hostName;\ remotePort;\ sendText) \longrightarrow Integer$

Parameter	Type		Description
udp_ID	Longint	\rightarrow	UDP socket reference
hostName	String	\rightarrow	Name or IP address of server
remotePort	Integer	\rightarrow	Remote port to connect to (0=any)
sendText	Text	\rightarrow	Text to be sent

Function result Integer \leftarrow Error code

Description

The $\overline{UDP_SendTo}$ command can be used to send text data using the udp_ID socket.

hostName is the name or IP address of the server where the text will be sent.

remotePort is the number of the port to be connected to. If you pass 0, any available port will be used.

sendText is the text to be sent.

See Also

 $\underline{UDP_ReceiveFrom},\,\underline{UDP_SendBLOBTo}.$

 $UDP_ReceiveFrom (udp_ID; hostName; remotePort; text) \longrightarrow Integer$

Parameter	Type		Description
udp_ID	Longint	\rightarrow	UDP socket reference
hostName	String	\rightarrow	Name or IP address of server(s) from which to receive text
		\leftarrow	Name or IP address of server that responds
remotePort	Integer	\rightarrow	Remote port to connect to (0=any)
text	Text	\leftarrow	Text received

Function result Integer \leftarrow Error code

Description

The $\overline{UDP}_ReceiveFrom$ command can be used to receive text that is sent via the udp_ID socket.

hostName is the name or IP address of the server(s) where the text will be received. (You can pass either a specific IP address or a broadcast address). This parameter then receives the name or IP address of the server that responds.

 ${\it remotePort}$ is the number of the port to be connected to. If you pass 0, any available port will be used.

The text received is returned in the text parameter.

See Also

 $\underline{UDP_ReceiveBLOBFrom}, \underline{UDP_SendTo}.$

UDP_SendBLOBTo

version 11

 $UDP_SendBLOBTo \ (udp_ID; \ hostName; \ remotePort; \ sendBlob) \longrightarrow Integer$

Parameter	Type		Description
udp_ID	Longint	\rightarrow	UDP socket reference
hostName	String	\rightarrow	Name or IP address of server
remotePort	Integer	\rightarrow	Remote port to connect to (0=any)
sendBlob	BLOB	\rightarrow	BLOB to be sent

Function result Integer \leftarrow Error code

Description

The *UDP_SendBLOBTo* command can be used to send a BLOB using the *udp_ID* socket.

hostName is the name or IP address of the server where the BLOB will be sent.

remotePort is the number of the port to be connected to. If you pass 0, any available port will be used.

sendBlob is the BLOB to be sent.

See Also

 $\underline{UDP_ReceiveBLOBFrom}, \underline{UDP_SendTo}.$

UDP_ReceiveBLOBFrom

version 11

 $UDP_ReceiveBLOBFrom\ (udp_ID;\ hostName;\ remotePort;\ blob) \longrightarrow Integer$

Parameter	Type		Description
udp_ID	Longint	\rightarrow	UDP socket reference
hostName	String	\rightarrow	Name or IP address of server(s) from which to receive BLOB
		\leftarrow	Name or IP address of server that responds
remotePort	Integer	\rightarrow	Remote port to connect to (0=any)
blob	BLOB	\leftarrow	BLOB received

Function result Integer \leftarrow Error code

Description

The \overline{UDP} _ReceiveBLOBFrom command can be used to receive a BLOB that is sent via the udp_ID socket.

hostName is the name or IP address of the server where the BLOB will be received. (You can pass either a specific IP address or a broadcast address). This parameter then receives the name or IP address of the server that responds.

 ${\it remotePort}$ is the number of the port to be connected to. If you pass 0, any available port will be used.

The BLOB received is returned in the blob parameter.

See Also

 $\underline{UDP_ReceiveFrom}, \underline{UDP_SendBLOBTo}.$

version 11

 $UDP_Delete~(udp_ID) \longrightarrow Integer$

Parameter Type Description

udp_ID Longint \rightarrow UDP socket reference

Function result Integer \leftarrow Error code

Description

The *UDP_Delete* command can be used to close a UDP socket that has been previously opened using <u>UDP_New</u> by passing its reference in *udp_ID*.

See Also

UDP_New.



- Special Internet Commands, Overview
 NET Finger (hostName; searchText; results) → Integer
 NET Ping (hostName; text; alive {; timeout}) → Integer
 NET Time (hostName; date; time; offset) → Integer
 NET NameToAddr (hostName; ip_Longint) → Integer
 NET AddrToName (ip_Longint; hostName; ip_Address) → Integer
 NET Resolve (hostName; ipOrHost) → Integer

Other related commands:

- String -- 4D • Time string (seconds)
- <u>IT_SetTimeOut</u> (timeout) Integer -- Theme: IC Utilities



version 6.5

NET_Finger (hostName; searchText; results) → Integer

Parameter	Type		Description
hostName	String	\rightarrow	Host name or IP address
searchText	String	\rightarrow	Search text
results	Text	\leftarrow	Finger results

Function result Integer \leftarrow Error Code

Description

Given an IP address of a machine to search and the text of a user account name on the machine, the <u>NET_Finger</u> command will return the finger results into <u>results</u>. 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 his/her ".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 "4d.com", you could perform the search as:

```
$error:=NET_Finger ("www.4d.com";"johnt";$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 "4d.com" to perform a remote finger query of the user "johnt@4d.com".

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

 ${\it results}$ is the text returned which contains the results of the search.

NET Ping

version 6.8.1 (Modified)

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

Parameter	Type		Description
hostName	String	\rightarrow	Host name or IP address
text	Text	\rightarrow	Text to send in ping
alive	Integer	\leftarrow	1 = Alive, 0 = Timeout/Inactive
timeout	Integer	\rightarrow	# of seconds to wait, 0 = use IT_SetTimeOut
			value

Function result Integer \leftarrow Error Code

Description

The NET_Ping command 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 with the IT_SetTimeOut command.

 $\textbf{Note:} \ \ \text{This parameter is not taken into account under Windows 95/98 and Millennium.}$

See Also

IT_SetTimeOut.

NET_Time

version 6.5

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

Parameter	Type		Description
hostName	String	\rightarrow	Host name or IP address
date	Date	\leftarrow	Date
time	Longint	\leftarrow	Time, expressed as seconds since midnight
offset	Integer	\rightarrow	Hours to offset

Function result Integer ← Error Code

Description

Given the host name or IP address to an Internet Time server, the NET_Time command 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).

See Also

Time string

 $NET_NameToAddr (hostName; ip_Longint) \longrightarrow Integer$

Parameter	Type		Description
hostName	String	\rightarrow	Host name or IP address
ip_Longint	Longint	\leftarrow	Long Integer reference to the address

Function result Integer \leftarrow Error Code

Description

The NET_NameToAddr command 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.

 $NET_AddrToName~(ip_Longint;~hostName;~ip_Address) {\:\longrightarrow\:} Integer$

Parameter	Type		Description
ip_Longint	Longint	\rightarrow	Long Integer reference to the address
hostName	String	\leftarrow	Host name
ip_Address	String	\leftarrow	IP address

Function result Integer \leftarrow Error Code

Description

The NET_AddrToName command 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.

version 6.5

NET_Resolve (hostName; ipOrHost) → Integer

Parameter	Type		Description
hostName	String	\rightarrow	Host name or IP address
ipOrHost	String	\leftarrow	Returns the opposite value

Function result Integer \leftarrow Error Code

Description

Given a host name in the first parameter, the NET_Resolve command 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 <u>NET_Resolve</u> 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'
```

IC Utilities

Utility Commands, Overview

IT MacTCPInit → Integer

IT Platform → Integer

IT Ly Version → String

IT TCPversion (stackKind; stackVersion) → Integer

IT MacTCPVer (versionCode) → Integer

IT MacTCPVer (versionCode) → Integer

IT MacTCPVer (versionCode) → Integer

IT SetTimeOut (timeout) → Integer

IT SetTimeOut (timeout) → Integer

IT Encode (fileName; encodedFile; encodedMode) → Integer

IT Decode (fileName; decodedFile; decodeMode) → Integer

IT Decode (fileName; decodedFile; decodeMode) → Integer

IT GetProxy (protocol; proxyRind; proxyHostName; proxyPort; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyHostName; proxyPort; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyHostName; proxyPort; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyHostName; proxyPort; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyHostName; proxyPort; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyPost; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyPost; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyPost; proxyVost; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyPost; proxyPost; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyPost; proxyVost; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyPost; proxyVost; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyRind; proxyPost; proxyVost; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyRind; proxyPost; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyRind; proxyPost; proxyVost; proxyUserID) → Integer

IT SetTory (protocol; proxyRind; proxyR



IT_MacTCPInit

version 6.8.1 (Modified)

 $IT_MacTCPInit \rightarrow Integer$

Parameter Type Description

This command does not require any parameters

Function result Integer ← Error Code

Description

The IT_MacTCPInit command 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 if they are not dial-up scheme users. In this case, 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 (see IT_PPPStatus command).

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

See Also

IT_PPPConnect.

IT_Platform → Integer

Parameter Type Description

This command does not require any parameters

Platform Type (0 = 68K Code, 1 = PPC Code,

2 = Windows)

Description

The IT_Platform function 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)
```

version 6.5

 $IT_Version \longrightarrow String$

Parameter Type Description

This command does not require any parameters

Function result String \leftarrow Version String

Description

The *IT_Version* function 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 2004 (Modified)

IT_TCPversion (stackKind; stackVersion) -> Integer

Parameter	Type		Description
stackKind	Integer	\leftarrow	0 = None, 3 = WinSock, 4 = BSD
stackVersion	Text	\leftarrow	Version number of the TCP stack

Function result Integer \leftarrow Error Code

Description

The <u>IT_TCPversion</u> command returns information about the type of TCP stack currently in use by the 4D Internet Commands. The type of Stack varies by platform. Under Macintosh, only BSD is now 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 (Obsolete, see Compatibility note)
- 2 Open Transport (Obsolete, see Compatibility note)
- 3 WinSock
- 4 BSD Sockets

Compatibility notes:

- $\bullet \ \ MacTCP \ is \ no \ longer \ supported \ (starting \ from \ version \ 6.8). \ Therefore, \ the \ \textit{stackKind} \ parameter \ will \ no \ longer \ return \ "1".$
- $\bullet \ Open \ Transport \ is \ no \ longer \ supported \ (starting \ from \ version \ 2004). \ Therefore, \ the \ \textit{stackKind} \ parameter \ will \ no \ longer \ return \ "2".$

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 6.8.1 (Modified)

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

 $IT_MacTCPVer \ (versionCode) {\:\longrightarrow\:} Integer$

ParameterTypeDescriptionversionCodeInteger←Version Code of MacTCP installed

Function result Integer \leftarrow Error Code

Description

Starting with 4D Internet Commands version 6.8, MacTCP is no longer supported. Consequently, the versionCode parameter systematically returns 0 whatever the platform and/or OS used.

See Also

IT_TCPversion.

IT_MyTCPAddr

version 6.8.1 (Modified)

 $IT_MyTCPAddr~(ip_Address;~subnet) \longrightarrow Integer$

Parameter	Type		Description
ip_Address	String	\leftarrow	IP address of users machine
subnet	String	\leftarrow	Subnet Mask in IP form

Function result Integer \leftarrow Error Code

Description

The <code>IT_MyTCPAddr</code> command 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.

 $\textbf{Note:} \ \ \textbf{Under Windows 95, the } \textit{subnet} \ \textbf{parameter returns the default subnet mask for the address class specified by } \textit{ip_Address.}$

 $IT_SetTimeOut\ (timeout) \longrightarrow Integer$

Parameter Type Description

timeout Integer → Timeout in seconds; limited to 0 thru 127

Function result Integer ← Error Code

Description

The IT_SetTimeOut command sets the integer value of the timeout period in seconds. This value is limited to between zero and 127 seconds. By default, the timeout period is 30 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_MsqInfo

POP3_MsgLstInfo

POP3_GetMessage

POP3_MsgLst

POP3_Download

POP3_VerifyID

POP3_UIDToNum

IMAP_Login

IMAP_VerifyID

IMAP_Capability

IMAP_ListMBs

IMAP_SubscribeMB

IMAP_GetMBStatus

IMAP_SetCurrentMB

IMAP_Delete

IMAP_MsgInfo

IMAP_MsgLstInfo

IMAP_GetMessage

IMAP_MsqLst

IMAP_SetFlags

IMAP_GetFlags

IMAP_MsqFetch

IMAP_Download

IMAP_CopyToMB

IMAP_CreateMB

IMAP_RenameMB

IMAP_DeleteMB

NET_Finger

NET_Ping



IT_GetTimeOut

version 6.5

 $IT_GetTimeOut \ (timeout) {\:\longrightarrow\:} Integer$

$\begin{array}{cccc} \textbf{Parameter} & \textbf{Type} & \textbf{Description} \\ \text{timeout} & \text{Integer} & \leftarrow & \text{Timeout seconds} \end{array}$

Function result Integer \leftarrow Error Code

Description

The <code>IT_GetTimeOut</code> command returns the current timeout value for the commands listed in <code>IT_SetTimeOut</code>. <code>timeout</code> is the current value in seconds of the timeout period.

See Also

IT_SetTimeOut.

IT_ErrorText

version 6.5

 $IT_ErrorText~(error) \longrightarrow String$

Parameter Type Description

error Integer → Error code returned from other commands

Function result String \leftarrow Text of the error

Description

The IT_ErrorText command 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
```

See Also

Appendix A, Programming Tips.

IT_Encode (fileName; encodedFile; encodedMode) -> Integer

Parameter	Type		Description
fileName	Text	\rightarrow	A LocalPath to a file
encodedFile	Text	\rightarrow	LocalPath file specification
		\leftarrow	Path to resulting encoded file
encodedMode	Integer	\rightarrow	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 *IT_Encode* command 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 ".hqx" 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.
- a full LocalPath folder specification (without specifying the file name) providing the folder in which the encoded file will be saved; the file name will be the original file name with a suffix defining the encoding mode.
- a null string (in this case, the IT_Encode 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 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.

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".



IT_Decode (fileName; decodedFile; decodeMode) -> Integer

Parameter	Type		Description
fileName	Text	\rightarrow	LocalPath to an encoded file
decodedFile	Text	\rightarrow	LocalPath file specification
		\leftarrow	Path of decoded file
decodeMode	Integer	\rightarrow	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 \leftarrow Error Code

Description

The IT_Decode command 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.

- a full LocalPath file specification providing a name and location for the decoded file.
- a full LocalPath folder specification indicating the folder that will receive the decoded file using the original file name.
- a null string (in this case, the IT_Decode command) will provide its own name for the document, located 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 Apple Double (decode Modes~4~&~6), this command looks for a file named "% filename" for the resource fork.

See Also

FTP MacBinary, FTP Receive, IT Encode



IT_GetProxy

version 6.8.1 (Modified)

 $IT_GetProxy \ (protocol; proxyKind; proxyHostName; proxyPort; proxyUserID) \longrightarrow Integer$

Parameter	Type	Description
protocol	Integer \rightarrow	1 = FTP; 2 = SMTP; 3 = POP3; 4 = IMAP
proxyKind	Integer ←	0 = None; 1 = SOCKS
proxyHostName String		Host name or IP address of the SOCKS Proxy host
proxyPort	Integer ←	Proxy port to connect to
proxyUserID	Text ←	UserID for SOCKS
proxyKind proxyHostNam proxyPort	Integer ← e String ← Integer ←	0 = None; 1 = SOCKS Host name or IP address of the SOCKS Prohost Proxy port to connect to

Function result Integer \leftarrow Error Code

Description

Given a specified protocol, the IT_GetProxy command returns the current settings being used by the 4D Internet Commands related to the routage of the specified protocol. The values will be at their default state unless a prior call to IT_SetProxy altered the settings. For a complete description of the parameters, see IT_SetProxy.

protocol is an integer value that specifies the protocol to examine. A value of 1 will indicate FTP protocol. A value of 2 will indicate SMTP protocol. A value of 3 will indicate POP3 protocol. A value of 4 will indicate IMAP protocol.

proxyKind returns the current settings determining if a SOCKS proxy host is used. A value of 1 routes all requests for the specified protocol through the specified SOCKS Host. A value of zero does not route requests for the specified protocol through any SOCKS Host.

proxyHostName returns the current settings determining the Host Name or IP address of the SOCKS Proxy host in use.

 $proxyPort\ returns\ the\ current\ settings\ determining\ the\ port\ number\ used\ for\ the\ specified\ protocol\ to\ communicate\ with\ the\ SOCKS\ Proxy\ host.$

 $\ensuremath{\textit{proxyUserID}}$ returns the current settings determining the user ID.

See Also

IT_SetProxy.

IT SetProxy

version 6.8.1 (Modified)

IT_SetProxy (protocol; proxyKind; proxyHostName; proxyPort; proxyUserID) -> Integer

Parameter	Type		Description
protocol	Integer	\rightarrow	1 = FTP; 2 = SMTP; 3 = POP3; 4 = IMAP
proxyKind	Integer	\rightarrow	0 = None; 1 = SOCKS
proxyHostNan	ne String	\rightarrow	Host name or IP address of the SOCKS Proxy
proxyPort	Integer	\rightarrow	Proxy port to connect to
proxyUserID	Text	\rightarrow	UserID for SOCKS

Function result Integer Error Code

Description

The <u>IT_SetProxy</u> command allows you to negociate a connection using the specified protocol and then send all further requests through the SOCKS Host (SOCKS Proxy). If you are just connecting to an intranet, then you will probably not have to communicate through the SOCKS Host. However, it all depends on how your company has their firewall set up. The <u>IT_SetProxy</u> settings have an interprocess scope and effect all the connections using the specified protocol in any 4D process.

Note: Socks (or "SOCKS") is a protocol that a proxy server can use to accept requests from client users in a company's network so that it can forward them across the Internet. If your workstation is located behind a firewall and you want to access an information located on the Internet, the SOCKS host receives your request, forwards the request through the firewall, and then returns the information to your client application.

protocol is an integer value that specifies the protocol to be routed through the specified SOCKS Proxy host. A value of 1 will effect FTP protocol. A value of 2 will effect SMTP protocol. A value of 3 will effect POP3 protocol. A value of 4 will indicate IMAP protocol.

proxyKind is an integer value indicating whether the specified protocol should be routed through a SOCKS Proxy host or not. A value of 1 will route all requests for the specified protocol through the specified SOCKS Host. A value of zero won't route requests for the specified protocol through any SOCKS Host.

proxyHostName is the Host name or the IP address of the SOCKS Proxy machine.

proxyPort is an integer value that specifies the port to use for the specified protocol to communicate with the SOCKS Proxy host.

proxyUserID is a text value that identifies the user. The user ID is given by your network administrator. proxyUserID can be an empty text ("").

Example

Using the following method, all FTP connections will be routed through the specified SOCKS Proxy Host.

 $\textbf{Note:} \ \ \text{For clarification purposes, this example does not contain error checking.}$

The following statement stops routing FTP connections through any SOCKS Proxy Host.

```
$err:=IT_SetProxy (1;0;$proxyAdd;$proxyPort;"")
```

See Also

IT_GetProxy.

IT_GetPort

version 6.8.1 (Modified)

 $IT_GetPort (protocol; port) \longrightarrow Integer$

Parameter	Type		Description
protocol	Integer	\rightarrow	1 = FTP; 2 = SMTP; 3 = POP3; 4 = IMAP
port	Integer	\leftarrow	Port Number

Function result Integer \leftarrow Error Code

Description

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

See Also

Appendix B, TCP Port Numbers, IT_SetPort.

IT_SetPort

version 6.8.1 (Modified)

 $IT_SetPort\ (protocol;\ port) {\:\longrightarrow\:} Integer$

Parameter	Type		Description
protocol	Integer	\rightarrow	1 = FTP; 2 = SMTP; 3 = POP3; 4 = IMAP
port	Integer	\rightarrow	Port Number

Function result Integer \leftarrow Error Code

Description

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

See Also

Appendix B, TCP Port Numbers, IT_GetPort.

 $IT_PPPConnect\ (pppProfil) {\:\longrightarrow\:} Integer$

Parameter Type pppProfil String

Description

→ Dial-up name = Null string on Mac OS, fill under Windows

Function result Integer \leftarrow Error code

Description

The IT_PPPConnect command opens the current dial-up connection under Mac OS or the specified dial-up connection (pppProfil) parameter under Windows. This command acts as a function and returns an integer value error if the connection cannot be opened.

This command must be executed each time you need to execute a set of Internet Commands that works online. On completion, you must execute IT PPPDisconnect to close the current connection.

PPP (Point-to-Point Protocol) is a protocol for communication between two computers using a serial interface, typically a personal computer connected by a phone line to a server. For instance, your Internet server provider may supply you with a PPP connection so that the provider's server can respond to your requests, pass them on to the Internet and forward your requested Internet responses back to you. Essentially, it packages your computer's TCP/IP packets and forwards them to the server where they can actually be served on the Internet.

PPP is usually preferred over the former de facto standard Serial Line Internet Protocol (SLIP) because it can handle synchronous as well as asynchronous communication. PPP can share a line with other users and has an error detection function, neither of which is true for SLIP. If a choice is possible, PPP is preferred.

See Also

 $\underline{\mathsf{IT_MacTCPInit}}, \underline{\mathsf{IT_PPPDisconnect}}, \underline{\mathsf{IT_PPPStatus}}.$

 $IT_PPPDisconnect\ \{(pppProfil)\} \longrightarrow Integer$

Parameter Type pppProfil String

Description

 → Dial-up name = Null string on Mac OS, optionally filled under Windows

Function result Integer \leftarrow Error code

Description

The IT_PPPDisconnect command closes the current dial-up connection previously opened by IT_PPPConnect.

pppProfil is a text value specifying the dial-up connection to close.

Under Windows, this parameter may be useful when several PPP connections are opened simultaneously. Using this parameter will ensure good running whatever the user network configuration.

Under Windows:

- If only one connection is opened and pppProfil is not passed or is passed as a null string, IT_PPPDisconnect closes the opened connection.
- If several connections are opened and pppProfil is not passed or is passed as a null string, IT_PPPDisconnect returns an error and does not close any connection.
- $\bullet \ \, \text{If } \textit{pppProfil} \ \, \text{is passed and valid, the specified connection is closed whatever the number of opened connections.}$

Under Mac OS

This parameter is not taken into account.

See Also

IT_PPPConnect, IT_PPPStatus.

 $IT_PPPS tatus \ \{(pppProfil)\} \longrightarrow Integer$

ParameterTypeDescriptionpppProfilString→ Dial-up name = Null string on Mac OS, optionally filled under Windows

Function result Integer \leftarrow 1 if connected; 0 if connecting; -1 if error

Description

The IT_PPPStatus command allows you to check the status of a connection opened with the IT_PPPConnect command or opened manually.

pppProfil is a text value specifying which opened connection to check.

Under Windows, this parameter is optional but may be useful to ensure good running whatever the user network configuration.

Under Windows:

- If pppProfil is passed and valid, specified connection status is return.
- If pppProfil is not passed or is passed as a null string, IT_PPPStatus will return:
- · -1 if several connections are opened,
- the status of the opened connection if only one connection is opened.

Under Mac OS:

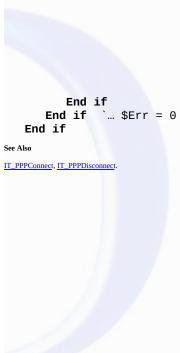
This parameter is not taken into account.

IT_PPPStatus returns an integer denoting the connection status. It returns:

- 1 if connected,
- 0 if connecting
- $\ensuremath{\raisebox{.3ex}{\text{\circ}}}$ -1 in case of a connection failure or if not connected.

Example

```
`Method GetMessages (this method is executed in a process)
If(mPPPConnect($vPPPProfil; 120))
   $vErrCode:=IT MacTCPInit
   If($vErrCode=0)
      $vErrCode:=POP3 Login...
   F1se
      ALERT("Connection failed")
   End if
End if
   `Method mPPPConnect
C BOOLEAN($0) `returns True if we are currently connected, False if connection failed
C TEXT($1) `null string if Mac OS, Entry Name if Windows
C_INTEGER($2) `timeout in seconds
If (IT_PPPStatus =1)
   $0:=True `we are already connected
Else
   $vTimeoutLength:=$2
   $vTimeout:=False
   $vErr:=IT PPPConnect($1)
   If($vErr=0)
      $vStart:=Current time
      Repeat
         DELAY PROCESS(Current process;30)
         $vStatus:=IT_PPPStatus($1)
         $vTimeout:=((Current time-$vStart)>$vTimeoutLength)
      Until (($vStatus=1) | $vTimeout) `we are connected or time out
      If(Not($vTimeout))
         $0:=True `we are connected
```





version 6.5

Executing Commands via a Case Statement

In many of the examples in this document, 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:

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:

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 to not of the commands fails. If every command executes successfully, 4D will proceed down to run the Else condition where we set the SSentOK flag to indicate that the message was composed and sent successfully.

Suggestions when auto-replying to POP3 or IMAP 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;
- 443 HTTPS.

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

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

^{• 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...)

 $[\]bullet \ 49152 \ to \ 65535 \ (Dynamic \ and/or \ Private \ Ports): The \ Dynamic \ and/or \ Private \ Ports \ are \ free \ of \ use.$

115	Simple File Transfer Protocol
118	SQL Services
119	Network News Transfer Protocol
123	Network Time Protocol
129	Password Generator Protocol
143	Interactive Mail Access Protocol v2
144	NewS
150	SQL-NET
171	Network Innovations Multiplex
172	Network Innovations CL/1
201	AppleTalk Routing Maintenance
202	AppleTalk Name Binding
203	AppleTalk Unused
204	AppleTalk Echo
205	AppleTalk Unused
206	AppleTalk Zone Information
207	AppleTalk Unused
208	AppleTalk Unused
213	IPX
396	Novell Netware over IP
407	Timbuktu
443	Secured protocol
531	chat
532	readnews
533	for emergency broadcasts
540	uucpd
541	uucp-rlogin
565	whoami
600	Sun IPC server
767	phone
888	AccessBuilder
	118 119 123 129 143 144 150 171 172 201 202 203 204 205 206 207 208 213 396 407 443 531 532 533 540 541 565 600 767



All 4D Internet Commands (with the exception of <u>IT ErrorText</u> & <u>IT Version</u>) 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 or WinSock network layer

Zero No Error

Error 1 -> 61 BSD network layer

Error >= 10000 4D Internet Commands Error

4D Internet Commands Error Codes

If 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.
- 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.
- FTP 552 Requested file action aborted. Exceeded storage allocation (for current directory or dataset).
- 10063 FTP 553 Requested action not taken. File name not allowed.
- 10064 No response has been received within the given timeout period.
- 10065 Not an FTP file.
- 10066 Error in processing Base64.
- 10067 Error in processing AppleSingle.
- 10068 Error in processing Quoted-Printable.
- 10069 FTP session was closed by the server.
- 10070 Not an FTP directory.
- 10071 TCP session was closed by the server
- 10072 Invalid encode kind
- 10073 Invalid decode kind
- 10074 An asynchronous DNR call did not complete
- 10075 An asynchronous OpenTransport call did not complete
- 10076 OpenTransport bind failed
- 10077 OpenTransport connect failed
- 10078 Maximum MacTCP streams reached
- 10079 Error in processing uuencode
- 10080 Cannot load ICMP library
- 10081 Error in processing MacBinary
- 10082 MacBinary checksum error

10083	Could not open a file
10084	No FTP information received
10085	Unknown FTP information received
10086	Proxy connection failed
10087	Standard file I/O error
10088	FTP reentrant error
10089	SLI.DLL is not loaded
10091	Error trying to make an IMAP connection
10092	A maibox is not selected
10093	Invalid message part
10094	Error with IMAP LOGIN
10095	Error with IMAP LOGOUT
10096	Error with IMAP CAPABILITY
10097	Error with IMAP SELECT
10098	Error with IMAP FETCH
10099	Error with IMAP PARTIAL
10100	Error with IMAP STORE
10101	Error with IMAP EXPUNGE
10102	Error with IMAP SEARCH
10103	Error with IMAP COPY
10104	Error with IMAP CREATE
10105	Error with IMAP DELETE
10106	Error with IMAP RENAME
10107	Error with IMAP SUBSCRIBE
10108	Error with IMAP UNSUBSCRIBE
10109	Error with IMAP LIST
10110	Error with IMAP LSUB
10111	Error with IMAP STATUS
10112	Error with IMAP CLOSE
10113	Error with AUTHENTICATION
BSD Error Codes	
1	Operation not permitted
4	Interrupted system call

13	Permission denied
14	Bad address
22	Invalid argument
24	Too many open files
35	Operation would block
36	Operation now in progress
37	Operation already in progress
38	Socket operation on non-socket
39	Destination address required
40	Message too long
41	Protocol wrong type for socket
42	Protocol not available
43	Protocol not supported
44	Socket type not supported
45	Operation not supported
46	Protocol family not supported
47	Address family not supported by protocol family
48	Address already in use
49	Can't assign requested address
50	Network is down
51	Network is unreachable
52	Network dropped connection on reset
53	Software caused connection abort
54	Connection reset by peer
55	No buffer space available
56	Socket is already connected
57	Socket is not connected
58	Can't send after socket shutdown
60	Operation timed out
61	Connection refused
WinSock Error Coc	les

-10004 Blocking call cancelled

-10013 Permission denied

-10014	Bad address	
-10022	Invalid argument	
-10024	No more sockets av	ailable
-10035	Non-blocking socke	et would block
	progress	nction invoked while a blocking function is in
-10037	An attempt was made already completed	de to cancel an asynchronous operation that has
-10038	Specified socket des	scriptor 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	loes not match the other parameters in the call
-10042	Protocol option is u	nknown or invalid
-10043	Specified protocol i implementation	s not supported by the Windows Sockets
-10044	Specified socket type	be is not supported by the specified address family
-10045	Socket does not sup	port the specified operation
-10046	Protocol family not	supported
-10047	Specified address fa implementation or	amily is not supported by the Windows Sockets
	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 ne	etwork subsystem
-10051	Network cannot be	reached from this host at this time
-10052	Connection was dro	pped and must be reset
-10053	Connection was abo	orted because of a timeout or other error condition
-10054	Connection was res	et by the remote host
-10055	Windows Sockets in provided in an API	mplementation is out of buffer space or the space
	call by the applicati	on 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
	s 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 velopers may find this list useful if they are writing their own mail communication procedures using low-level TCP commands.
211	System status, or system help reply
214	Help message [Information on how to use the receiver or the meaning of a particular
	non-standard command; this reply is useful only to the human user]
220	<domain> Service ready</domain>
221	<domain> Service closing transmission channel</domain>
250	Requested mail action okay, completed
251	User not local; will forward to <forward-path></forward-path>
354	Start mail input; end with <crlf>.<crlf></crlf></crlf>
421	<pre><domain> Service not available, closing transmission channel [This may be a reply to</domain></pre>
	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] Syntax error in parameters or arguments 501 502 Command not implemented Bad sequence of commands 503 Command parameter not implemented 504 Requested action not taken: mailbox unavailable [e.g., mailbox not 550 found, no access] User not local; please try <forward-path> 551 552 Requested mail action aborted: exceeded storage allocation Requested action not taken: mailbox name not allowed [e.g., mailbox 553 syntax incorrect] Transaction failed 554 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. Restart marker reply. In this case, the text is exact and not left to the 110 particular implementation; it must read: MARK yyyy = mmmmWhere yyyy is User-process data stream marker, and mmmm server's equivalent marker (note the spaces between markers and "="). Service ready in nnn minutes. 120 125 Data connection already open; transfer starting. File status okay; about to open data connection. 150 200 Command okay. 202 Command not implemented, superfluous at this site. System status, or system help reply. 211 212 Directory status. 213 File status. Help message on how to use the server or the meaning of a particular 214 non-standard command. This reply is useful only to the human user. NAME system type. Where NAME is an official system name from the 215

list in the

Assigned Numbers document.
Service ready for new user.
Service closing control connection. Logged out if appropriate.
Data connection open; no transfer in progress.
Closing data connection. Requested file action successful (e.g., file transfer or file
abort).
Entering Passive Mode (h1,h2,h3,h4,p1,p2).
User logged in, proceed.
Requested file action okay, completed.
"PATHNAME" created.
User name okay, need password.
Need account for login.
Requested file action pending further information.
Service not available, closing control connection. This may be a reply to any command if
the service knows it must shut down.
Can't open data connection.
Connection closed; transfer aborted.
Requested file action not taken. File unavailable (file busy).
Requested action aborted: local error in processing.
Requested action not taken. Insufficient storage space in system.
Syntax error, command unrecognized. This may include errors such a command line
too long.
Syntax error in parameters or arguments.
Command not implemented.
Bad sequence of commands.
Command not implemented for that parameter.
Not logged in.
Need account for storing files.
Requested action not taken. File unavailable (e.g., file not found, no access).
Requested action aborted: page type unknown.

Requested file action aborted. Exceeded storage allocation (for current 552 directory or dataset). Requested action not taken. File name not allowed. 553

Appendix D, Additional Information...

version 6.8.1 (Modified)

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 or Internet Explorer.

<u>http://www.internic.net/</u>: To understand what a Domain Name is 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 an RFC is 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.

<u>http://www.w3c.org/</u>: All you need to know about the World Wide Web.

http://www.imap.org: Site reserved for IMAP protocol. You will find all the useful information concerning this protocol there.

- Appendix A, Programming Tips
- Appendix B, TCP Port Number
- Appendix C, 4D Internet Commands Error Codes
- Appendix D, Additional Information...

Downloading Mail, Overview

- File Transfer, Overview
 FTP Append (ftp_ID; localPath; hostPath; progress) → Integer
 FTP ChangeDir (ftp_ID; directory) → Integer
 FTP Delete (ftp_ID; hostPath) → Integer
 FTP GetDirList (ftp_ID; directory; names; sizes; kinds; modDates{; modTimes})) → Integer
 FTP GetDirList (ftp_ID; directory; names; sizes; kinds; modDates{; modTimes})) → Integer
 FTP GetPassive (ftp_ID; passiveMode) → Integer
 FTP GetPassive (ftp_ID; passiveMode) → Integer
 FTP Login (hostName; userName; password; ftp_ID{; welcomeText}) → Integer
 FTP Login (hostName; userName; password; ftp_ID{; welcomeText}) → Integer
 FTP MakeDir (ftp_ID; directory) → Integer
 FTP MakeDir (ftp_ID; directory) → Integer
 FTP PrintDir (ftp_ID; directory) → Integer
 FTP Progress (left; top; windowTitle; thermoText; cancel) → Integer
 FTP Receive (ftp_ID; hostPath; localPath; progress) → Integer
 FTP Rename (ftp_ID; hostPath; newPathName) → Integer
 FTP Send (ftp_ID; passiveMode) → Integer
 FTP SetType (ftp_ID; ftp_Mode) → Integer
 FTP VerifyID (ftp_ID) → Integer

Glossary and Terminology

• Low Level Routines, Overview

- MSG_Charset (decodeHeaders; bodyCharset) → Integer
 MSG_Delete (fileName{; folder}) → Integer
 MSG_Extract (fileName, decode; attachmentPath; enclosureList) → Integer
 MSG_FindHeader (fileName; headerLabel; headerValue) → Integer
 MSG_GetBody (fileName; offset; length; bodyText) → Longint
 MSG_GetHeaders (fileName; offset; length; rawText) → Integer
 MSG_GetMessage (fileName; offset; length; rawText) → Integer
 MSG_GetBody (fileName; offset; length; rawText) → Integer
 MSG_HasAttach (fileName; attachCount) → Integer
 MSG_MessageSize (fileName; headerSize; bodySize; msgSize) → Integer
 MSG_SetPrefs (stripLineFeed; msgFolder; attachFolder) → Integer
 MSG_SetPrefs (stripLineFeed; msgFolder; attachFolder) → Integer

- NET AddrToName (ip_Longint; hostName; ip_Address) → Integer
 NET Finger (hostName; searchText; results) → Integer
 NET NameToAddr (hostName; ip_Longint) → Integer
 NET Ping (hostName; text; alive; (imeout) → Integer
 NET Resolve (hostName; ipOrHost) → Integer
 NET Time (hostName; date; time; offset) → Integer

- Parameter Formats (hostName; ip_LongInt; mailAddress; addressList; localPath; hostPath; tcp_ID; smtp_ID; pop3_ID; imap_ID; ftp_ID) → Integer POP3_BoxInfo (pop3_ID; msgCount; msgSize) → Integer POP3_Charset (decodeHeaders; bodyCharset) → Integer POP3_Delete (pop3_ID; startMsg; endMsg) → Integer POP3_DowNLoad (pop3_ID; msgNumber; headerOnly; fileName) → Integer POP3_GetMessage (pop3_ID; msgNumber; offset; length; msgText) → Integer POP3_GetMessage (pop3_ID; msgNumber; offset; length; msgText) → Integer POP3_GetMessage (pop3_ID; msgNumber; offset; length; msgText) → Integer POP3_Logout (pop3_ID) → Integer POP3_Logout (pop3_ID) → Integer POP3_Logout (pop3_ID) → Integer POP3_MsgLnfo (pop3_ID; start, end; hdrArray; msgNumArray; idArray; valueArray) → Integer POP3_MsgLstInfo (pop3_ID; startMsg; endMsg; sizeArray; msgNumArray; idArray) → Integer POP3_Reset (pop3_ID) → Integer POP3_Reset (pop3_ID) → Integer POP3_SetPrefs (stripLineFeed; msgFolder{; attachFolder}) → Integer POP3_SetPrefs (stripLineFeed; msgFolder{; attachFolder}) → Integer POP3_UIDToNum (pop3_ID; uniqueID; msgNumber) → Integer POP3_VerifyID (pop3_ID) → Integer POP3_VerifyID (pop3_ID) → Integer POP3_VerifyID (pop3_ID) → Integer POP3_VerifyID (pop3_ID) → Integer POP3_POP3_VerifyID (pop3_ID) → Integer POP3_VerifyID (pop3_ID) → Integer POP3_VerifyID

• Receiving Mail, Overview

- Sending Mail, Overview

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

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

 SMTP_Auth (smtp_ID; userName; password{; authMode}) → Integer

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

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

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

 SMTP_C (smtp_ID) → Integer

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

 SMTP_C (smtp_ID) → Integer

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

 SMTP_D Comments (smtp_ID; comments{; deleteOption}) → Integer

- SMTP Comments (smtp_ID); comments {; deleteOption}) → Integer

 SMTP Date (smtp_ID; smgDate; msgTime; timeZone; offset{; deleteOption}) → Integer

 SMTP Encrypted (smtp_ID; encrypted{; deleteOption}) → Integer

 SMTP Encrypted (smtp_ID; encrypted{; deleteOption}) → Integer

 SMTP From (smtp_ID; msgFrom{; deleteOption}) → Integer

 SMTP GetPrefs (lineFeeds; bodyType; lineLength) → Integer

 SMTP Host (smtp_ID; hostName{; deleteOption}) → Integer

 SMTP InReplyTo (smtp_ID; inReplyTo{; deleteOption}) → Integer

 SMTP Kewwords (smtp_ID; kewwords{; deleteOption}) → Integer

 SMTP New (smtp_ID) → Integer

 SMTP OuickSend (hostName; msgFrom; msgTo; subject; message) → Integer

 SMTP References (smtp_ID; replyTo{; deleteOption}) → Integer

 SMTP Send (smtp_ID) → Integer

 SMTP Send (smtp_ID); msgSender{; deleteOption}) → Integer

 SMTP Sender (smtp_ID; msgSender{; deleteOption}) → Integer

 SMTP Subject (smtp_ID; subject{; deleteOption}) → Integer

 SMTP Subject (smtp_ID; subject{; deleteOption}) → Integer

 SMTP Subject (smtp_ID; msgTo{; deleteOption}) → Integer

 SMTP To (smtp_ID; msgTo{; deleteOption}) → Integer

- TCP Close (tcp_ID) → Integer
 TCP Listen (remoteHost; localPort; remotePort; timeout; tcp_ID) → Integer
 TCP Open (hostName; remotePort; tcp_ID{; sessionSettings}) → Integer
 TCP Receive (tcp_ID; text) → Integer
 TCP ReceiveBLOB (tcp_ID; blobToReceive) → Integer
 TCP Send (tcp_ID; sendText) → Integer
 TCP SendBLOB (tcp_ID; blobToSend) → Integer
 TCP State (tcp_ID; statusCode) → Integer

- UDP Commands, Overview

 UDP _Delete (udp_ID) → Integer

 UDP _New (localPort; udp_ID) → Integer

 UDP _ReceiveBLOBFrom (udp_ID; hostName; remotePort; blob) → Integer

 UDP _ReceiveFrom (udp_ID; hostName; remotePort; text) → Integer

 UDP _SendBLOBTo (udp_ID; hostName; remotePort; sendBlob) → Integer

 UDP _SendTo (udp_ID; hostName; remotePort; sendBlob) → Integer

 UDP _SendTo (udp_ID; hostName; remotePort; sendText) → Integer

 Utility Commands, Overview